TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Information Technology

MUHAMMAD FARHAN

AUTOMATED CLUMP SPLITTING FOR BIOLOGICAL CELL SEGMENTATION IN MICROSCOPY USING IMAGE ANALYSIS

Master of Science Thesis

Examiners: Professor Olli Yli-Harja
Dr. Antti Niemistö
Examiners and topic approved in the Computing and Electrical Engineering Faculty Council meeting on 04 November 2009

Preface

This Master's thesis work is carried out in the Department of Signal Processing, Tampere University of Technology, Finland. First of all, I would like to thank The Almighty Allah for giving me the courage and knowledge to do this research work. Secondly, I pay my deepest gratitude to Professor Olli Yli-Harja for providing me with the opportunity to work in the vibrant and diversified group of Computational Systems Biology and also for examining this work. Next, I would like to thank my mentor and supervisor, Dr. Antti Niemistö, from the depth of my heart for not only introducing me to this topic but also for his guidance, support and advice throughout. It could not have been done without his support, supervision and patience with my work, especially with my writing. I am also thankful to Dr. Pekka Ruusuvuori and Dr. Jyrki Selinummi for their positive feedback.

I am highly indebted to the people and Government of Finland for providing me with the opportunity to study here and perform this research work. I am also thankful to all the people of the Computational Systems Biology group for providing me with such kind of conducive environment for the research work, especially my office colleagues for their kind support and helping hands.

I would also like to thank all my friends in Finland and everywhere else. It was due to their constant moral support and encouragement that I am able to perform this work. Special thanks to all my Pakistani friends in Tampere whose nice company, support and get-togethers never let me feel lonely and away from my home.

Last but not least, I would like to pay my deepest gratitude to my loving family especially my parents whose kindness, care, love and encouragement helped me during my studies here and made me able to do this work.

Tampere, Finland, September 23, 2010.

Muhammad Farhan

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
Farhan, Muhammad: Automated Clump Splitting for Biological Cell Segmentation in Microscopy Using Image Analysis.
Master of Science Thesis, 60 Pages.
September 2010
Major: Signal Processing
Examiners: Prof. Olli Yli-Harja and Dr. Antti Niemistö
Keywords: cell, clump splitting, image analysis, concavity point, cell segmentation, split line.

Formation of clumps due to touching or overlapping of individual objects in an image is common. The process is natural in some cell cultures, for instance, yeast cells typically grow in clumps. Automated analysis of images containing such clumps requires the capability to split them into their constituent objects. Failure of the segmentation methods to split the clumps leads to the requirement of developing clump splitting methods to be used as post-processing step towards overall segmentation. The goal of this thesis work is to study and develop an automated method for splitting cell clumps in images of biological cells. To achieve this goal we studied previous clump splitting methods found in the literature. One of the best methods is based on defining split lines by detecting and linking concavity points. We found that this method has deficiencies in it and first modified it to achieve improved clump splitting results. We also developed a novel method for clump splitting following a similar approach.

Like any other concavity point-based clump splitting method, both these methods start with finding all the concavity points on the contour of the clumps. Contrary to the original method, these methods look for every possible valid concavity point in a concavity region using curvature analysis, thus minimizing false split lines as well as under-segmentation. The modified method then uses Delaunay triangulation to narrow down the list of all the possible split lines between all the concavity points to a list of candidate split lines. Finally, it uses a set of features such as saliency and alignment to define a cost function. The best split line is found for each concavity point yielding the minimum value for the cost function. On the other hand, the novel method uses variable size rectangular window to search for the concavity point-pairs forming the split lines. This makes the method less dependent on user-defined parameters. We also propose some

post-processing steps that remove some non-cellular objects based on *a priori* information on cell shapes.

We compared the performance of these two methods with the performance of the original method and of a widely used method that is based on the watershed transform. Three different sets of images of yeast cells were used. Precision and recall analysis was used to show that the two methods proposed in this thesis outperform the two methods taken from the literature. Although the targeted application of the methods is splitting of cell clumps, it can be applied to split clumps of other convex objects as well.

Table of Contents

List of Symbols and Abbreviations

A, B	Sets representing binary image and structuring element respectively
A_i	Area of the region i
b	number of bits
$b(.)$	gray-scale structuring element function
B_i	Boundary segment
$\beta(.)$	boundary extraction function
β_{ij}	angle between directional vector and split line
c_i	Weight coefficients
$c(j)$	Concaveness measure at point j of contour
C	Convexity measure
$C(.)$	convex hull
C_i	Concavity point
C_{i1}, C_{i2}	End points of the convex hull chord
χ	Cost function
CA_i	Concavity angle
CC_{ij}	Concavity-concavity alignment
CD_i	Concavity depth
CD_m	Maximum concavity depth
CD_n	Second largest concavity depth of the clump
CF_{ij}	Cost function for split line between concavity points i and j
CL_{ij}	Concavity-line alignment
CR	Concavity ratio
$\binom{\cdot}{\cdot}$	Combination or choice number
D_e	Euclidean distance measure

DT	Delaunay triangulation
e_{ij}	Split line between concavity point *i* and *j*
$\in$	Element of
EMST	Euclidean minimum spanning tree
$f(.)$	gray-scale image function
FM	F-measure
FN	False negative
FP	False positive
$g(.)$	Morphological gradient function
g	Gradient of intensity along a certain path
h	Height or length of window
$h(.)$	Top-hat transformation function
HSI	Hue, Saturation, Intensity color space
$i(.)$	Illumination function
$\cap$	Intersection operator
k	Curvature value at a contour point
K_i	Convex hull chord
L_{lchord}	Maximum length chord in a cluster
λ_i	Threshold values
LMC	Local maximum curvature
M_j	5x5 window centered at point *j*
$M(f)$	Multi-scale morphological gradient
N	Total number of gray-levels
NA	Numerical aperture
$\oplus$	Dilation operator
$\ominus$	Erosion operator
$\circ$	Opening operator
$\bullet$	Closing operator

$\circledast$	Match operator
P	Boundary point in the direction of directional vector
$P_{min}(a,b)$	Minimum number of pixels between a and b along region contour
p	Diameter of hypothetic circumference of perimeter
$ø_i$	Angle between split lines
$Ø$	Empty set
$ø_i$	Sum of angles of tangents on contour of i_{th} partition
PR	Precision
r (.)	Reflectance function
RC	Recall
RGB	Red, Green, Blue color space
$\wedge$	Reflection about origin operator
s(.)	Skeleton extraction function
SA_{ij}	Saliency
SCMD	Saccharomyces cerevisiae morphological database
$\sum$	Summation operator
$\subseteq$	Subset operator
SVM	Support vector machine
T	Intensity level threshold
T_i	Tangent to contour at point i
τ_i	Threshold values
θ	Angle measure
θ^*	Final set of split lines
TP	True positive
2-D	Two dimension
3-D	Three dimension
u_{ij}	Directional vector from concavity point i to j
$\cup$	Union operator

v_i	Directional vector
V	Set of concavity points
w	Width of the window
x	Angle between the directional vector and reference vector
(x,y)	Spatial co-ordinates
z	Set of displacements of structuring element

Chapter 1

Introduction

While performing image analysis in different domains it is often observed that the objects in the image form dense clusters or clumps. Manual detection of such clumps and their separation into their constituent objects can be relatively easy, assuming that the person who is doing the manual analysis possesses some prior knowledge about the objects in the image. However, if the same task needs to be performed for a large number of images, splitting of the clumps needs to be done automatically using some computer-based algorithm. This is often a difficult task due to the nature of the clumps present in the images. Nevertheless, an accurate automatic splitting of clumps is often of paramount importance in terms of extracting accurate information from the images [18].

Generally, clumps are formed either due to touching or overlapping of objects with each other. In the case of biological cell cultures, cells tend to grow in such a way that they form clumps, such as growing of the bud from mother cells in yeast. In addition, when there is a large density of cells in a particular area of the image or if the cells in the image are too close to each other, then due to optical projections the individual cells seem to overlap with each other therefore forming a clump [29]. Moreover, the process of preparing samples for future analysis and preserving the cells from being decayed as well as the varying behavior of individual cells on different stimuli contribute to the formation of cell clumps [39].

Resolving individual objects from these clumps using general image segmentation methods or some basic morphological operations, such as erosion, is generally not possible. Even in cases where the segmentation of the image into foreground and background pixels is easily achieved because of high contrast between them, segmentation often fails to separate the individual objects from clumps. This may be because of the fact that the grey-level values of the objects forming the clumps often have a high degree of resemblance among themselves. Due to this reason, some specific clump splitting methods are needed which will give high efficiency with low number of over- and under-

segmented objects. These clump splitting methods are usually applied to the binary segmented images as a post-processing step towards overall segmentation.

Splitting of clumps is essential in a wide range of applications in the field of computer vision ranging from biological to industrial applications [3, 10, 43]. In tasks related to microscopic images of cells that have clumped together, the requirement is to split the cell clumps into individual cells automatically so that some further biological analysis can be performed on single cells [24, 35, 41]. In industrial applications the task may be to scan and detect individual objects transported on a conveyor belt [3]. These objects may vary in size and shapes and often overlap with each other. Accurate detection and hence the subsequent analysis of those objects depends heavily on the accurate splitting of those clumps into the constituent objects.

Construction of an automated clump splitting method which gives absolutely precise results for all images in even a small image data set still remains to be a challenging task. Given a particular image, it is quite easy to develop an algorithm which will accurately split all the clumps present in that image. However, when it comes to doing this automatically for a large image set with varying cell features, it is quite difficult to achieve the desired results. It is typical that clump splitting method require different parameters for different images in the data set.

There are three major approaches which are common in clump splitting methods. They are defined briefly as follows:

1. **Concavity point-based analysis:** When merging or overlapping of two or more convex-shaped objects occurs, the resultant object is concave, and the points of contact on the boundaries of the two objects are called concavity points. In concavity point-based analysis, see for example [10, 18, 39], first the concavity points in the clump are found. Then, the split lines are found by joining two concavity points provided that certain conditions are met. These methods depend heavily on how the decision of whether or not a split line is defined between two concavity points is made. Many of the methods found in the literature have proven unsatisfactory in practical applications.

2. **Mathematical morphology-based analysis:** This includes methods based on basic morphological operations as well as methods based on the morphological watershed transformation, see for example [26, 14, 34]. These methods are also unsatisfactory because in practical applications they tend to produce over-segmentation and under-segmentation, especially, when the objects vary a lot in size and shape.

3. **Model based or parametric fitting-based analysis:** In this type of analysis some model is fitted on the image data, for example, ellipse fitting is used [6, 17]. Me-

thods from this approach also need to find the concavity points to divide the contour into segments on which ellipse fitting is performed. The problem with these methods is that they are often computationally complex and involve a large number of parameters.

This thesis work is undertaken in order to solve the problem of splitting clumps in images of biological cells. Therefore further discussion primarily concentrates on the issues related to the clumps of biological cells and the ways to address them. The rest of the thesis is organized in the following way:

Chapter 2 provides the readers with the basic knowledge of digital image analysis. The focus is in briefly describing the image processing algorithms which are going to be used in the subsequent chapters. A review of clump splitting methods found in the literature is presented in Chapter 3. Chapter 4 discusses modifications that were made to gain some improvements in one of the methods described in Chapter 3. Moreover, a novel clump splitting method is also presented. Chapter 4 concludes with a presentation of post-processing steps that can be used to improve initial clump splitting results. The results that we obtained from the modified method as well as from the novel clump splitting method are presented in Chapter 5. A qualitative as well as quantitative comparison of these methods with selected methods from the literature is also included. Finally, Chapter 6 concludes the thesis along with discussing possible directions of future work.

Chapter 2

Fundamentals of Digital Image Analysis

This chapter provides the reader with a basic understanding of digital images and image analysis. The concepts that are presented here will be repeatedly used in the subsequent chapters. We start with a discussion on digital images and their representation. Next we move on to describe some basic concepts of morphological image processing. We conclude our discussion by presenting some of the approaches used for image segmentation. It is worth to mention here that most of this chapter is adopted from [1, 13, 32].

2.1 Digital Image

In the process of image acquisition, the imaging system forms an image by capturing a part of the illumination coming from the source that is reflected from the scene. Thus the image can be described by a 2-D function dependent on the illumination of the source and the reflectance of the scene element being imaged and is given by [13]

$$f(x,y) = i(x,y)\, r(x,y)\,, \tag{2.1}$$

where x and y are the 2-D spatial variables denoting the spatial coordinates of the image, $i(x,y) \in [0,\infty)$ is the illumination function, and $r(x,y) \in [0,1]$ is the reflectance function. The value of $f(x,y)$ at any point in space is always positive and corresponds to the intensity of light at that point in the scene [13].

When an image is acquired, it may be continuous both in space and in intensity. Digital processing of these images is only possible once they are sampled and quantized. These processes are performed to discretize both the spatial coordinates as well as the intensity values. The sampling of the image can simply be perceived as placing a rectangular grid on top of the image, whereas quantization is the process of representing the intensity values on those locations in terms of a real number representing one of finite number of intensity levels.

As soon as an image is digitized, it becomes possible to represent it by using an *m* x *n* matrix with *m* rows and *n* columns. Each element of this matrix is generally called *picture element* or *pixel* [13]. The amplitude value at each element of the matrix is proportional to the intensity of the light at that spatial point in the image and the value generally lies in the range of 0 to 2^b-1 for *b* bit images. This defines the gray-level resolution of an image. The larger the value of *b*, the higher the gray-level resolution of the image. On the other hand, spatial resolution is generally defined by the number of pixels in the image and is given by *m* x *n* number of pixels. The quality of the image depends on both the gray-level and the spatial resolutions.

A digital image can be binary, gray-scale or color image. The pixels in a binary image are represented by two intensity levels either 0 or 1. If the image is a gray-scale image, which has just the luminance intensity information but not the color information, then every pixel has a certain value of intensity in the range described above. However, the color images are envisaged as 3-D functions where there is a third dimension as well. This dimension is defined depending on the model used, such as Red Green and Blue (RGB) model, Hue Saturation and Intensity (HSI) model, for specifying color images. For example, in the RGB model, the third dimension is the color components which consist of *Red*, *Green* and *Blue* channels and all the colors are formed by the additive combination of these basic RGB colors. Every color component can be thought of and can be processed as an individual gray-scale image, and then laid on top of each other to form a color image. A pixel in a color image is thus composed of *b* x *3* number of bits and so the matrix used to represent the image is *m* x *n* x *3* in size.

In digital images, a set of pixels which are all connected to each other by a connectivity rule is called connected component. The pixels in these components generally have small variation in intensity among themselves whereas large variations from other groups of pixels thus giving rise to different objects [31]. Those pixels which are not a part of objects are called background pixels. These concepts are used while labeling the objects in the image for further analysis.

2.2 Morphological Image Processing

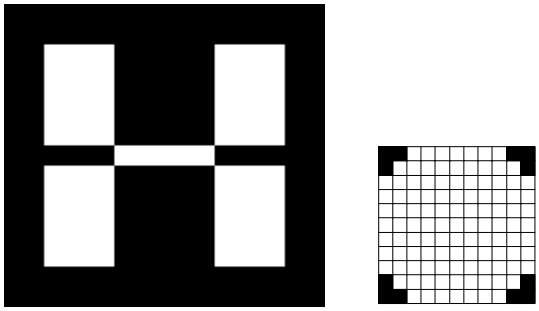
Mathematical morphology uses set theory to define, represent and analyze objects in digital images. Thus, in morphological image processing, set theory is used as a tool to determine the features for representing the shape of the region along with the features describing that representation in such a form so as to allow further processing. Apart from this, morphological image processing also offers some nonlinear filtering techniques, typically used in pre- and post-processing steps in image analysis [13]. Morphological filters have the property that they are increasing and idempotent. Increasing im-

plies that they are order preserving, whereas idempotence means that at one stage further successive iterations do not change the signal anymore [1].

Morphological operations are performed using a particular set in 2-D or 3-D integer space called structuring element. It can be of any size and shape depending on what it is used for. For instance, it can be a disk shaped element to process circular shaped objects in an image. It works similar to the window in filtering operations, sliding on the image with its center placed at the current pixel to be processed [13]. A structuring element can be considered as non-flat or flat based on whether or not it assigns some weight to different pixels of the window. Next, we discuss morphological image processing by first explaining the basic operations and then moving on to describe some of the basic algorithms of morphological image processing.

2.2.1 Basic Morphological Operations

The two main operations that are the basic building blocks in morphological image processing are dilation and erosion. They are used in many of the algorithms found in morphological image processing. Erosion and dilation are those morphological operations which do not have the property of idempotence [1]. Along with them the two other basic and important morphological operations that are frequently used are opening and closing. Extensions of them are close-opening and open-closing in which these operations are performed in the respective orders. We created an image, as shown in Figure 2.1(a), for illustrating the results of the basic morphological operations when applied on that. We used an 11 x 11 flat disc-shaped structuring element of neighborhood 6 as shown in Figure 2.1(b).



(a) (b)

Figure 2.1: (a) *Binary test image of size 220x230 pixels. (b) 11x11 flat disc-shaped structuring element with neighborhood 6.*

2.2.1.1 Dilation and Erosion

In the dilation operation, the structuring element is first flipped about its origin and then the origin is moved across the image pixels, and all the image pixels below the origin of

the structuring element are turned bright if the overlap between the structuring element and the image is not an empty set. Mathematically it can be defined as

$$A \oplus B = \left\{ z \mid \left(\hat{B}\right)_z \cap A \neq \emptyset \right\}, \tag{2.2}$$

where $\oplus$ is the symbol for the dilation operation, A and B are the sets representing the image and the structuring elements respectively, and z is the set of all displacements of the structuring element. The effect of dilation is that it expands the objects in the image as well as fills the small holes and openings in the image [13]. Figure 2.2 illustrates the dilation operation on the test image of Figure 2.1.

Figure 2.2: *Dilation. Result of dilation on test image of Figure 2.1.*

Erosion has the opposite effect of dilation on image pixels. In erosion, the origin of the structuring element is moved across the image pixels, and only those image pixels are kept to be bright, where, if the origin of the structuring element is placed and the whole structuring element resides inside the image object. Mathematically it can be written as

$$A \ominus B = \{ z \mid (B)_z \subseteq A \}, \tag{2.3}$$

where $\ominus$ is the symbol for the erosion operation. Erosion basically narrows or shrinks the image objects in addition to removing very small objects or thin parts of the objects [13]. Figure 2.3 shows the impact of erosion on the test image.

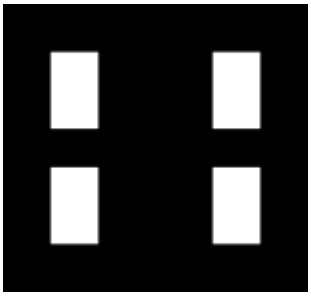


Figure 2.3: *Erosion. Result of erosion on test image of Figure 2.1.*

In gray-scale morphology, dilation is the process in which the maximum intensity value among the image pixels underlying the structuring element is given to the image pixel located below the origin of the structuring element. Similarly, erosion in gray-scale morphology picks the minimum intensity value from the image pixels under the structuring element and puts this value on the image pixel lying under the origin of the structuring element.

2.2.1.2 Opening and Closing

Even though dilation and erosion are complementary to each other, they do not perfectly reverse each other's action. Due to this reason, their successive application is order dependent, that is, the order in which they are applied on an image matters in the final output [13]. Depending on the order in which these two operations are performed, two other basic morphological operations are obtained, namely opening and closing. Closing operations are extensive, that is, the output of a signal is greater than the signal itself at a particular point in space. On the other hand, openings are anti-extensive, that is, output of a signal is smaller than the signal itself at a particular point in space [1].

In opening, the image is first eroded with a particular structuring element followed by dilation of the resultant image with the same structuring element. This way narrow bridges or small connections between objects as well as thin portions of objects are eliminated [13]. It also smoothes the object contour, for example, smoothens sharp edges as well as removes protrusions. It can be expressed mathematically as

$$A \circ B = (A \ominus B) \oplus B \,, \tag{2.4}$$

where $\circ$ represents the morphological opening. The result of opening the test image of Figure 2.1 is shown in Figure 2.4(a).

Closing, on the other hand is opposite to opening in which the image is first dilated then erosion is performed on the dilated image with the same structuring element. It basically plugs the gap between broken contour elements and also removes small holes in objects [13]. Similar to opening, it also smoothes the contour of the objects especially removes the inner edges, also shown in Figure 2.4 (b). Mathematically, it can be expressed as

$$A \bullet B = (A \oplus B) \ominus B \,, \tag{2.5}$$

where $\bullet$ represents the morphological closing. The effect of the closing operation on the test image of Figure 2.1 is shown in Figure 2.4(b).

In gray-scale morphology, opening and closing are obtained by using gray-scale erosion and dilation. Gray-scale opening removes the small light details in the image due to

application of the erosion before dilation. On the other hand, gray-scale closing removes dark details in the image due to application of the dilation before erosion [13].

(a) (b)

Figure 2.4: *Opening and Closing. Result of (a) opening and (b) closing on the test image of Figure 2.1.*

2.2.2 Morphological Image Processing Algorithms

In this part of the chapter we will discuss some of the algorithms for morphological image processing that are used in this thesis.

2.2.2.1 Boundary Extraction

The boundaries of the objects in an image are extracted by first eroding the input image A by a suitable structuring element B and then subtracting the resultant image from the original image, mathematically written as

$$\beta(A) = A - (A \ominus B), \tag{2.6}$$

where $\beta(A)$ is the boundary of set A. The thickness of the extracted boundary depends upon the size of the structuring element being used. Figure 2.5 shows the boundary of the input image extracted by using morphological operators.

(a) (b)

Figure 2.5: *Boundary extraction. (a) An object from an image and (b) its extracted boundary.*

2.2.2.2 Convex Hull

Finding the convex hull of a binary object is useful in many image analysis tasks. For a set to be convex, a line between any two points that belong to the set must be completely within the set. If the object is denoted by the set *A* then its convex hull is the smallest convex set *C* such that *A* is completely contained in *C*. The algorithm for finding the convex hull uses four structuring elements B^i of different types as depicted in Figure 2.6(a). The convex hull of an object *A* is given by

$$C(A) = \cup_{i=1}^{4} D^i \ , \tag{2.7}$$

where

$$D^i = {X^i}_{conv} \ ,$$

and

$${X^i}_k = \left(X_{k-1} \circledast B^i\right) \cup A, \tag{2.8}$$

where i = 1,2,3,4 and k = 1,2,3,.. , also ${X^i}_0 = A$ is taken as the starting point and convergence point is reached if for a particular value of i, ${X^i}_k = {X^i}_{k-1}$. The symbol $\circledast$ is used for finding the match ("hit") of the second set in the first set. An image object and its convex hull are shown in Figure 2.6(b) and (c) respectively.

Figure 2.6: *Finding the convex hull of a binary object. (a) Four different structuring elements. (b) An object from an image of size 180x240 pixels and (c) its convex hull.*

2.2.2.3 Skeleton Extraction

The skeleton of an image object is defined as a one pixel thick line going through the centre of the object such that it has equal distance from the object boundaries on either side. It can be obtained by iteratively peeling the object by using erosion or opening until there remains a thickness of one pixel. The selected structuring element should ensure that the topology of the region is retained in the process [37].

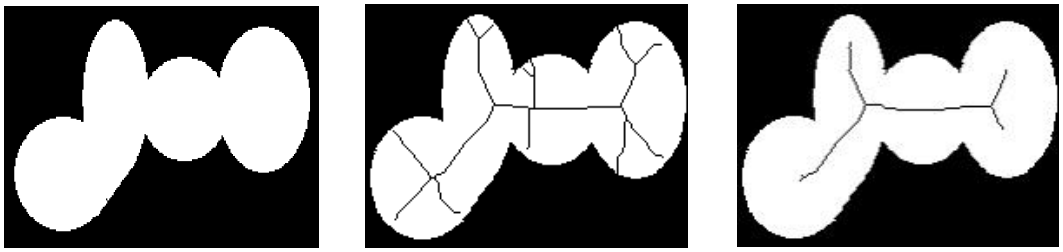
The skeletons of the objects in a binary image are obtained by

$$S(A) = \bigcup_{k=0}^{K} S_k(A) \ , \tag{2.9}$$

and

$$S_k(A) = (A \ominus k\,B) - (A \ominus k\,B) \circ B \ , \tag{2.10}$$

where $(A \ominus k\,B)$ is k-iterative erosions of A by B until K is reached such that one more erosion would make the output an empty set. Figure 2.7 demonstrates the operation of skeleton extraction.



(a) (b) (c)

Figure 2.7: *Skeleton Extraction. (a) An object from an image, (b) its extracted skeleton with spurious branches, and (c) skeleton without spurious branches.*

2.2.3 Morphological Operations on Gray-Scale Images

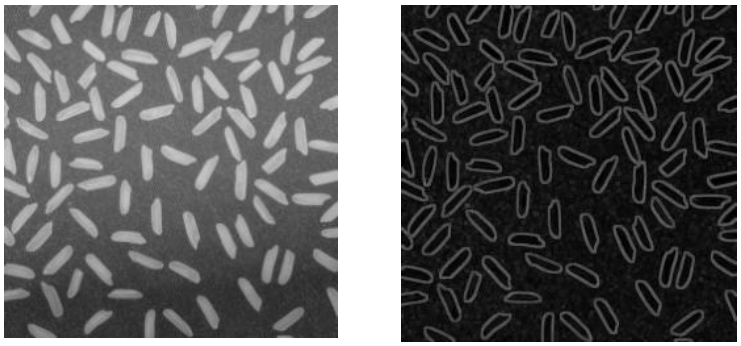
In Section 2.2.1, we discussed the gray-scale version of the basic morphological operations such as erosion, dilation, opening and closing. Here we discuss some of the other morphological operations performed on gray-scale images. These operations are used either to enhance the image details or to extract some important features from them.

2.2.3.1 Gradient

Gradient operations are used to find the sudden variations in intensity values among the pixels of a gray-scale image. The morphological gradient operation is applied on images to emphasize these intensity variations in addition to enhancing the details [13]. The subtraction of the eroded image from the dilated image gives the gradient, mathematically written as

$$g = (f \oplus b) - (f \ominus b) \, , \tag{2.11}$$

where f is the gray-scale image and b is the structuring element. Lower case denotes that these are functions in gray-scale morphology rather than the sets that are used in binary morphology. Figure 2.8 illustrates the morphological gradient operation.



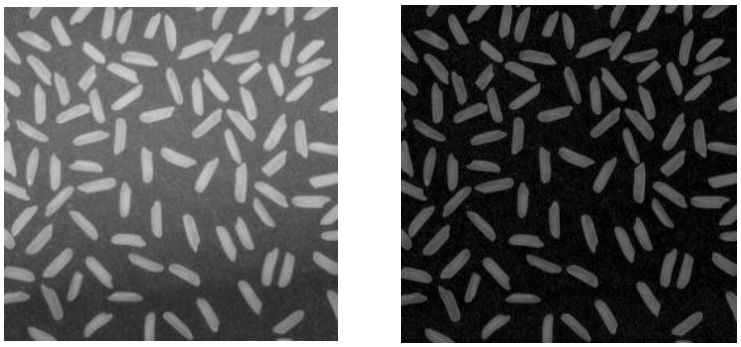
(a) (b)

Figure 2.8: *Image gradient. (a) Original gray-scale image and (b) its gradient.*

2.2.3.2 Top-Hat Transformation

Top-hat transform is a morphological transformation which magnifies the details in the regions of image where the contrast is low. It also emphasizes the objects which are darker than their surroundings [27] as illustrated in Figure 2.9. It is simply obtained by subtracting the morphologically opened image from the original image, expressed as

$$h = f - (f \circ b). \tag{2.12}$$



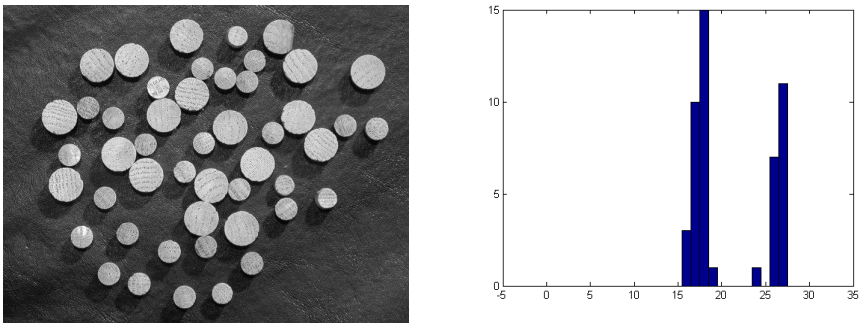
(a) (b)

Figure 2.9: *Top-Hat transformation. (a) Original gray-scale image from [12] and (b) image after application of top-hat transformation.*

2.2.3.3 Granulometry

Sometimes it is necessary to know the sizes of the objects in the image in order to proceed towards the right direction in the image analysis. One example of the use of this size distribution is in finding the optimal size of the structuring element for subsequent morphological operations [26]. Granulometry is the morphological technique to get the size distribution of the objects present in a gray-scale image. The idea is that we open an image with a particular sized structuring element which removes all the image objects smaller than the size of the structuring element. Taking the difference of this image from the original one we get the image containing objects removed from original after

opening. We can then deduce how many objects with size comparable to this structuring element were initially there in the image. The process is applied iteratively with increasing sizes of the structuring element. Later on the differences are normalized and a histogram is obtained which gives the distribution of the size of the objects in the image. Figure 2.10 shows an image containing objects of mainly two different sizes as is evident by two peaks in its size distribution found using granulometry.



(a) (b)

Figure 2.10: *Granulometry. (a) Original gray-scale image with varying sizes of objects and (b) its size distribution histogram using granulometry.*

2.3 Image Segmentation

Segmentation is the process in which an image is divided into certain segments or regions based on some similarity or common characteristics among the pixels of each region. The aim is to get such a representation of the image that makes it easier to perform further analysis on the image. It is one of the basic and necessary steps in image analysis, which can mean separating the objects in the image from the background or, in a more general framework, segregating distinct individual objects from all the objects. Within the context of this thesis, the former definition of segmentation is applicable. The real importance of segmentation is realized when some quantitative analysis is to be performed on the image. Then it is of utmost importance to have the objects absolutely separated from each other as well as from the background to perform further analysis successfully [13]. Segmentation is not straightforward; often the objects do not have apparent boundaries perhaps due to lack in sharp intensity transition between them and the background. Some prior knowledge about a few basic features of the image objects such as size, shape, and gray-level intensity do provide significant amount of information for the segmentation of those objects [32]. One example of taking image intensity into account is by assuming it as the height in the image. So the objects would be thought of as mountains, due to their high intensities, separated by valleys in an intensity landscape [40]. In that case, segmentation is achieved by locating those mountains in

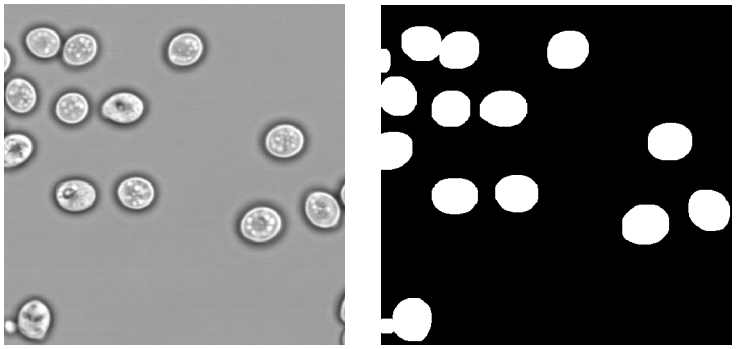
the landscape. One can find methods that use one or more of the above mentioned features for segmentation.

There are two main approaches used for image segmentation, one finds similarities among pixels of a certain region to segment the image into different regions and the other detects the discontinuities or sudden changes in the image intensity, for example, edges and boundaries of objects, to segregate the image into segments [13]. Thresholding and region based segmentation are two common methods and are based on the former approach. Morphological watershed segmentation is also a commonly used method for not only segmenting objects from background but also partitioning touching or overlapping objects from each other [36]. Here we briefly describe thresholding and watershed segmentation algorithm to end the chapter.

2.3.1 Thresholding

Thresholding is one of the basic and the most natural ways to segment an image into foreground and background pixels, i.e., into a binary image. This approach is useful in the case when there is a high degree of similarity among the object pixels as well as among the background pixels. Basically, the idea behind this approach is to find a threshold value T of intensity so that all the pixels with intensity value below T are marked as background pixels whereas others are marked as foreground pixels. Apart from using just a single value for threshold there can be a case in which more than one threshold values are required to successfully perform the segmentation. Such type of thresholding is often called multi-level thresholding [13]. There are different ways to find an appropriate threshold T. Perhaps the simplest one is finding the valleys in the histogram of intensity values such that the intensity values at the valleys are the segmentation threshold values [13].

Based on the manner in which T is obtained, we have three different types of thresholding namely global, local and adaptive thresholding. In global thresholding the threshold values are found globally, that is, using the intensity values of the whole image. So the same threshold value is used for the whole image. Local thresholding takes into account the intensity values in the neighborhood of a certain pixel to find the threshold values. Therefore, different thresholds are selected for different parts of the image. Finally, adaptive thresholding is also a kind of local thresholding but it involves the spatial coordinates as well and adaptively thresholds the different regions in the image [13]. Figure 2.11 shows the result of thresholding on the given gray-scale image.



(a) (b)

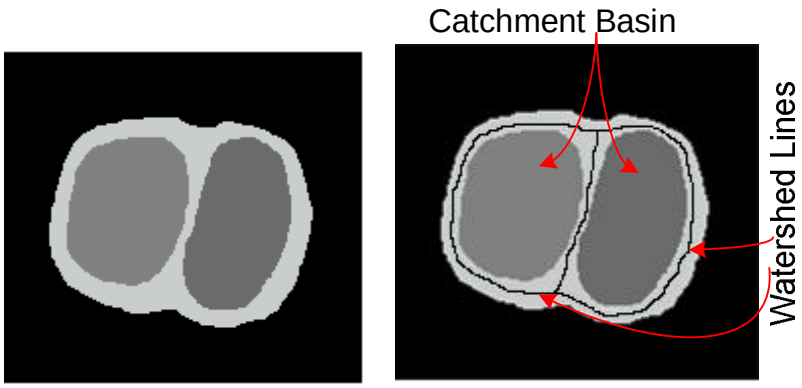
Figure 2.11: *Thresholding. (a) Original intensity image and (b) its binarized image after applying thresholding.*

2.3.2 Watershed Segmentation

The morphological watersheds or the watershed transform for image segmentation is basically a technique in which continuous boundaries, known as watershed lines, are found between the objects that may or may not be touching each other. The logic behind the name watershed comes from the concept in which the image is supposed to be composed of regions, formed by the objects, such that every region possesses its own intensity minimum. The *catchment basin* of that minimum is the set of points, on which if a drop of water is placed, it would end up falling to that point of minimum. The desired *watershed lines* are the locus of all those points, on which if a drop of water is placed, it can fall in any of the points of minimum that are adjacent to it [13].

Practically, this partitioning is realized by supposing that there is a hole punched in every region of a minimum, and water is rising into the regions from below at a uniform rate. Then there will be a point in time when the water from one region tends to overflow in the other region thus trying to merge them together. However, a dam is constructed which restrains the water from doing so. Seen from the top, the boundaries of the top of the dam would be visible that are analogous to the desired segmentation lines or the watershed lines [13]. Figure 2.12 depicts the watershed segmentation applied on the original image to get the image objects separated by watershed lines.

Practically, there exist some false minima in the images which lead to oversegmentation when the watershed transformation is applied directly [32]. In order to solve this problem, often marker-controlled watershed transformation is used. The idea is to use some features to obtain markers corresponding to the regions in the image. These markers are then used as the minima for subsequent application of watershed segmentation [32].



(a) (b)

Figure 2.12: *Watershed transformation. (a) Original image. (b) Image after application of Watershed segmentation.*

Chapter 3

Review of Clump Splitting Methods

In this chapter we present a review of clump splitting methods found in the literature. As described briefly in Chapter 1, the clump splitting methods can be categorized into three different approaches: concavity point-based methods [10, 18, 19, 24, 37, 38, 39, 43], mathematical morphology-based methods [7, 14, 20, 26, 34] and model- based or parametric fitting based methods [2, 6, 17, 30, 42]. Here in this chapter, we choose to discuss only those methods, from all the three approaches, which were found to be the most effective ones while being novel in their technique. In addition, the types of images that we have had in our data set also influenced the selection of the methods to be reviewed.

Although the original image can also be used to perform clump splitting analysis, almost all the methods found in the literature assume that the images are binarized and discard the original intensity information. As already mentioned in Chapter 2, the results of the overall automated analysis hugely depend on how accurately the binarization is done.

3.1 Concavity Point-Based Methods

Concavity points are the points on the boundary of clustered convex objects which are formed due to touching, overlapping or merging of two or more objects. Basically these are the points that are identified as the points with high concaveness and high value of curvature [10, 18]. Figure 3.1 shows three objects merged together to form a clump and the points of contact at the boundary of the two objects are the concavity points, highlighted with white dots.

Concavity point-based methods are quite effective and well known for splitting of clumps in cell microscopy images. The reason behind these methods being popular is that they try to imitate the human approach of separating clumped objects by looking for some prominent points on the object contour and then drawing a line between those

point-pairs which satisfy a certain set of conditions. There are many different methods which vary in how those points are found in the images and how the split lines are chosen.

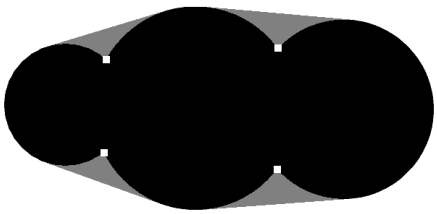


Figure 3.1: *Concavity points. (a) Clump of objects with its concavity points marked with white dots.*

3.1.1 Spatial and Gradient Parameter-Based Cell Segmentation

The clump splitting algorithm proposed by Fernandez et al. in [10] uses spatial and gradient parameters to find the line between concavity points for segregation of clumped objects in an image. The method was applied on the images containing clumps of plant cells. The decision about drawing a line between the two concavity points is influenced by two conditions: the distance between the two concavity points as compared to the perimeter of the clumped object and the flatness of the path between the concavity points.

The authors use the top hat transform, see Section 2.2.3.2, to enhance the contrast between the foreground cells and cell clumps from the background. After that, contours of the cells are extracted by using a morphological algorithm for boundary extraction given by Equation 2.6 in Section 2.2.2.1.

A concaveness measure is used to find the concavity point on the contour of the cell clumps. For every point j on the contour of the cell clumps the value of concaveness is found by

$$c(j) = \sum_{j-1}^{j+1} \sum_{x=1}^{5} \sum_{y=1}^{5} M_j \cap A, \tag{3.1}$$

where M_j is the 5x5 window centered at j and A is the binary image. For every contour pixel j, not only the foreground pixels in the 5x5 window centered at j but also the foreground pixels in the 5x5 window centered at the two adjacent contour pixels to j is taken into account to make the concaveness measure more robust. This way the concavity points along the contour would have a large value as compared to points on convex portions and on straight lines along the contour. Finally, thresholding is applied to the con-

caveness values, and pixels with high values of concaveness are found to be the concavity points in the cell clump. The idea is depicted in Figure 3.2 (b).

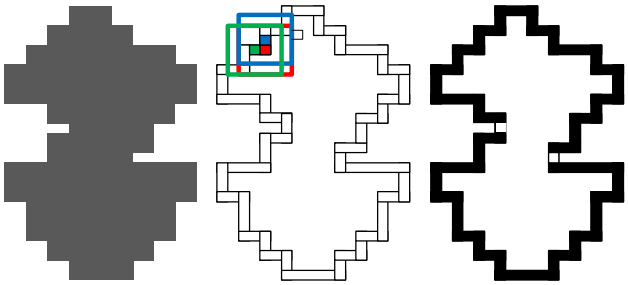


Figure 3.2: *Concaveness measure-based concavity point detection. (a) Image of cell clump. (b) Contour of cell clump with three different windows for evaluation of concaveness at a point. (c) Final image with concavity points marked with white points.*

Concavity point-pairs are then formulated so that a line can be drawn between them to split the cell clump into individual cells. The authors propose two parameters for finding concavity point-pairs which they referred to as the spatial parameter and gradient parameter.

The spatial parameter condition requires that the Euclidean distance between the two concavity points and the perimeter of the cell clump have a relationship given by

$$D_e < \frac{p}{\pi}, \tag{3.2}$$

where D_e is the Euclidean distance between two concavity points and p is the perimeter of cell clumps, that is, the number of contour points between those two concavity points. This condition makes sure that only those concavity points are joined by a line which arise due to overlapping of two cells and are not naturally present in the cells. Hence only for the former case the value of D_e will be smaller than the diameter of hypothetic circumference of p.

The gradient parameter takes into account the gray-level intensity values along the line joining two concavity points and is given by

$$g = \sum_{n=2}^{i} |Y_n - Y_{n-1}|, \tag{3.3}$$

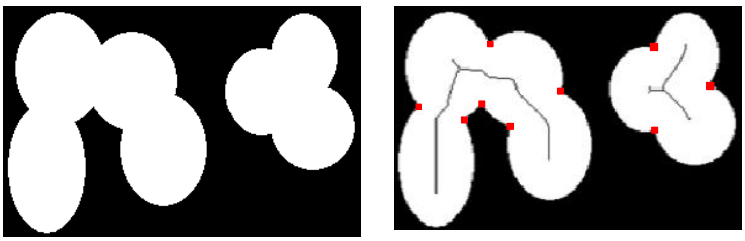
where i is the number of pixels in the line and Y_n is the gray-level intensity value at the n^{th} pixel of the line. It is required to have a minimum $grad$ value for a line to be a candidate line. A threshold value for the gradient is proposed to be 2*N, where N is total

number of gray-levels in the image, and it is expected that a candidate line gives a gradient value less than this threshold to make the concavity point-pair a valid one.

Each concavity point is examined against all the other concavity points to get the best pair for it, satisfying the above two conditions. Once the pairs are formed, they are joined by drawing a line in order to isolate the individual cells from the cell clump.

3.1.2 Form Analysis-Based Segmentation of Cell Clumps

Wang et al. presented a method in [37] for splitting of cell clumps in a microscopic image containing cells. The method uses form analysis to differentiate cell clusters from individual cells. Although the size and form of cells usually vary quite a lot in a particular image data set, it can still be assumed without much error that the cells have an elliptical form with not much difference in major and minor radius. Here in this method, a bounding polygon of a prototype form of the cell is fitted on the contour of the region under observation to examine its shape in order to separate single cells and cell clumps. Those regions which are not convex are identified as cell clumps and are therefore further processed so that they are splitted into single cells.



(a) (b)

Figure 3.3: *Form analysis-based cell segmentation. (a) Original image with cell clumps and (b) skeletonized image with concavity points found using minimum distance from the skeleton.*

After identification of cell clumps, they are skeletonized, see Section 2.2.2.3 for details. Typically the contours of the objects are affected by noise and they need to be smoothed, because otherwise skeletonization may lead to some unnecessary branches. Moreover, some short branches that are not consistent with the topology of the underlying objects, referred to as parasitic components, are often found after skeletonization is done. A morphological algorithm for pruning is used to remove them. Figure 3.3 shows cells and cell clumps along with their skeletons.

In the next step the candidate points for split lines are found. This is done by using the information from the contour and the skeleton of the cell clumps. Each point on the con-

tour of each cell clump is taken and its minimum distance from the skeleton is found. Using this distance data along with the respective index of contour points, a distance histogram is formed which, after further processing with a band pass filter, gives alternating peaks and valleys where the valleys correspond to the candidate points for split lines. These candidate points form a list with distance values of every point listed against it. Finally thresholding is performed on the distance values to eliminate those points which cannot be regarded as concavity points. Figure 3.3 (b) shows the concavity points found using minimum distance from the skeleton.

The authors present a stepwise procedure which uses the found concavity points to get the split lines. The previously created list of concavity points is first sorted with respect to the decreasing distance value. Once the list is sorted, the concavity point with the largest distance measure is taken, and its possible partner concavity point is found by testing every other concavity point for certain conditions (defined below) to be met. After obtaining the first concavity point-pair, the partner concavity point for the point with the second largest distance is found, provided that that point was not already selected as a partner of the a concavity point. This process is iterated until a partner is found for all the concavity points. Once a point is selected as a partner for a split line, it is no more considered to get a partner of its own; however, it is possible that it can be a partner for more than one concavity points.

There are two major conditions regarding the construction of a split line, which are checked before verifying other conditions to narrow down the possible concavity point-pairs. First condition is that the split line should pass through the skeleton. This requires the concavity point pairs to be facing each other. The second condition is that the split lines should neither intersect each other nor should they pass through the background.

With the assumption that the cells have elliptical shape and have very little variation in their size and shape, the authors propose the following set of constraints for the selection of the partner for the given concavity point:

- The ratio of length of the split line and the minimum number of pixels between the two concavity points along the region contour should be less than a predefined threshold as given by

$$\frac{D_e(a,b)}{P_{\min}(a,b)} < \tau_1 , \tag{3.4}$$

  where a and b are the two concavity points, $D_e(a,b)$ is the Euclidean distance between a and b, $P_{min}(a,b)$ is the minimum number of pixels between a and b along the region contour, and τ_1 is a predefined threshold.

- Since the split line divides the region into two, therefore the second condition is that the ratio of the area of those two regions should satisfy

$$\frac{\max(A_1,A_2)}{\min(A_1,A_2)} < \tau_2 , \tag{3.5}$$

where A_1 and A_2 are the areas of the two regions after the split and τ_2 is the predefined threshold.

- The third condition is related to the length of the split line and is given by

$$D_e(a,b) < \tau_3 , \tag{3.6}$$

where τ_3 is a predefined threshold value.

- To qualify as a split line, the fourth condition is that if there are two parallel or almost parallel split lines, then the distance between them should be large in comparison to the maximum of the length of the two split lines as given by

$$\frac{\min(D_e(a,c),D_e(b,d))}{\max(D_e(a,b),D_e(c,d))} < \tau_4 , \tag{3.7}$$

where a, b and c, d are a pair of concavity point-pairs forming the two split lines, with a and c being on the same side of the skeleton and b and d on the other side, and τ_4 is a predefined threshold.

- The degree of parallelism is defined by

$$\Theta(L_{ab}, L_{cd}) < \tau_5 , \tag{3.8}$$

where Θ is the angle between line L_{ab} and line L_{cd} and τ_5 is a predefined threshold.

- Finally there is a condition on the ratio of the length of the split line to the length of the maximum chord of the clump defined by

$$\frac{D_e(a,b)}{L_{lchord}} < \tau_6 , \tag{3.9}$$

where L_{lchord} is the chord in the cluster with maximum length and τ_6 is a predefined threshold.

There may be a case that more than one concavity points qualify for being the candidate partner for the concavity point under consideration. In that case the point that gives the minimum length split line is chosen. A line is drawn between those concavity point-pairs and the same process is repeated until the whole cell clump is split into the constituent single cells.

3.1.3 Rule-Based Splitting of Clumps

In this method of splitting clumps, presented by Kumar et al. in [18], the authors propose certain rules to decide between the split and no-split classes. Instead of finding split lines directly, this method first finds the candidate split lines between the two concavity points, and from those candidate lines the best line is selected based on a cost function.

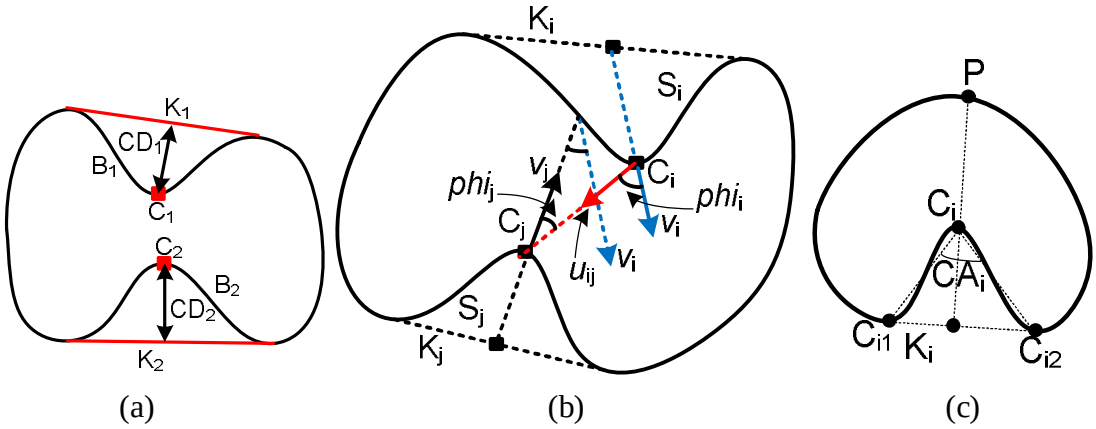


Figure 3.4: *Rule-based clump splitting. (a) A clump of objects showing concavity points* C_i*, convex hull chords* K_i*, and concavity depth features* CD_i*. (b) A clump of objects showing the directional vectors* v_i*,* v_j *and* u_{ij} *associated with the concavity points and the angles for calculation of* CC *and* CL *features. (c) A clump with only one concavity point defining the concavity angle* CA_i *to be used for finding a line between concavity point* C_i *and a boundary point* P*.*

The method starts from locating the concavity points. Concavity points C_i are defined as the points on the boundary segment which have the maximum perpendicular distance from their respective chords. For each concavity point, concavity regions are found and then their respective boundary segments B_i and convex hull chords K_i are evaluated as shown in Figure 3.4(a).

The authors propose some features that are used to narrow down a very large list of possible split lines into very few ones. The first feature is concavity depth CD_i, proposed by Rosenfeld in [25] and also shown in Figure 3.4(a). It is defined as the length of a perpendicular line from a concavity point to its convex hull chord. It gives a concave-

ness measure of a concavity point and is used to rule out those points which result from a noisy contour.

The first split line related feature is saliency SA_i. It is used to ensure that the concavity points constituting the split line have enough concaveness measure, that is, they are valid concavity points, and the distance between them is also minimal. It is given by

$$SA_{i,j} = \frac{\min(CD_i, CD_j)}{\min(CD_i, CD_j) + D_e(C_i, C_j)}, \tag{3.10}$$

where subscripts i and j refer to the two concavity points and $D_e(C_i, C_j)$ is the Euclidean distance measure. A large value for $0 < SA_{i,j} < 1$ is required to ensure the candidacy.

Naturally, two concavity regions, no matter how close they are, cannot share a split line unless they are aligned opposite to each other. This requirement is captured by two alignment features: concavity-concavity alignment CC_{ij} and concavity-line alignment CL_{ij}. A directional vector v_i towards the concavity point and originating from the midpoint of the corresponding convex hull chord is a parameter which defines the direction of the concavity region S_i and is used to find the alignment features, as shown in Figure 3.4(b). The angle between the directional vectors of the two concavity regions is used to find CC_{ij}, which indicates how much they are oppositely aligned, and is given by

$$CC_{ij} = \pi - cos^{-1}(v_i \cdot v_j), \tag{3.11}$$

where v_i and v_j are the two directional vectors. Ideally the angle between v_i and v_j should be equal to π and therefore CC_{ij} be equal to 0. The angles between the split line and the directional vectors of the corresponding concavity regions also tell us how much the regions are aligned. CL_{ij} is the feature which takes this into account and is defined by

$$\begin{aligned} CL_{ij} &= \max(\emptyset_i, \emptyset_j) \\ &= \max\left(cos^{-1}(v_i \cdot u_{ij}), cos^{-1}(v_j \cdot (-u_{ij}))\right), \end{aligned} \tag{3.12}$$

where $\emptyset_i$ and $\emptyset_j$ are the angles between the split line and the directional vectors v_i and v_j respectively, as depicted in Figure 3.4(b). To qualify for being a candidate it is required that CL_{ij} also has a small value, ideally equal to 0.

There are situations in clump splitting when a split line is needed between a concavity point and a boundary pixel on the other side of the concavity point. This situation arises when a concavity point is left without a pair. In that case the split line is formed between the concavity point C_i, the boundary pixel P and the midpoint of the correspond-

ing convex hull chord K_i, as shown in Figure 3.4(c). In such situations two features concavity angle *CA* and concavity ratio *CR* are used to determine if there should be a split line or not. They make sure that the concavity region is sharp, that the concavity region is deep, and that the region is the most concave of all the other concavity regions in the clump. They are defined as

$$CA = \angle C_{i1} C_i C_{i2} \,, \tag{3.13}$$

$$CR = \frac{CD_m}{CD_n} \,, \tag{3.14}$$

where C_i is the concavity point, C_{i1} and C_{i2} are the end points of convex hull chord, CD_m is the largest concavity depth of the clump, and CD_n is the second largest concavity depth. In the case that there is only one valid concavity point in the concavity region CD_n is replaced by the concavity depth threshold value. The split line is made if the concavity is sharp and large enough, that is, a low value of CA and a high value of CR is required for split.

Finally, to get the best split line of the chosen candidate lines, the authors propose a cost function given by

$$\chi = \frac{c_1 CD_i + c_1 CD_j + c_2}{D_e(C_i, C_j) + c_1 CD_i + c_1 CD_j + c_2} \,, \tag{3.15}$$

where c_1 and c_2 are the weights and are found by using a linear classifier such as the SVM classifier. The value of χ depends on how close to each other the concavity points are as well as on their concaveness. A large value of χ ensures a perfect split. The authors also observed that the decision boundary between the two classes, that is, split or no-split, is a straight line in 2-D feature space with $D_e(C_i, C_j)$ and $(CD_i + CD_j)$ as its two features.

The procedure is to recursively find the split lines between two concavity points satisfying the conditions stated above and, finally, if there still remains concavity points which could not get a pair then a split line between those concavity points and a boundary pixel is attempted so that in the end only single convex objects remain in the image.

3.1.4 Delaunay Triangulation-Based Splitting of Nuclei Clumps

This method for splitting clumps of cell nuclei is proposed by Wen et al. in [39]. It uses Delaunay triangulation for the formation of a reduced hypothesis space for the candidate split lines. This is followed by the application of certain geometrical constraints to re-

duce the size of this space, which after some inference rules gives the desired set of split lines.

The starting point of the method is finding the concavity points by defining the points having maximum curvature. The value of curvature is found at every point on the contour of the cells and cell clumps using the expression

$$k = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}}, \tag{3.16}$$

where x and y are the coordinates of the boundary pixels and the derivatives are found by using Gaussian derivative and convolving it with the boundary points. The list of values of k is then thresholded to get the points of local maximum curvature (LMC), that is, the concavity points. The set of such points is denoted by

$$V = \bigcup_{i=1}^{M} v_i ,$$

where v_i is the i^{th} point of the M total points of LMC.

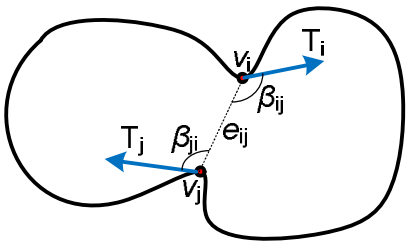


Figure 3.5: *Geometric attributes of concavity points as described in the method in [39].*

The line splitting the clumps are denoted by e_{ij}, with i and j being the indices of the two end points v_i and v_j of the edge respectively, as shown in Figure 3.5. For every split line there are features which are used during the application of geometrical constraints. The directional vectors of the tangent to the contour on those end points are denoted by T_i and T_j, whereas the angle between these directional vectors and the split line are denoted by β_{ij} and β_{ji} as shown in Figure 3.5.

A large number of split lines can be defined between M points, $\sum_{i=1}^{M-1} i$ to be exact, but most of them are invalid. To deal with this, Delaunay triangulation (DT) is quite effective. For a set of M points on a plane DT gives connected triangles between them such that not a single concavity point lies interior to the circumcircle of any of those triangles. DT is used here because of its properties that it discards intersecting split lines and also that its subgraph is a Euclidean minimum spanning tree (EMST). Moreover, the

property of DT that it maximizes the minimum of the interior angles of the triangles is desirable in this case.

The set of split lines obtained after DT is still quite large and also contains triangles. The fact, that triangles of edges are not applicable in most cases of split, suggests that still many of the edges need to be eliminated from the edge set. For that purpose geometrical constraints are applied which are as follows:

- The split line must be inside a clump and should neither pass through the background nor intersect the other split lines.

- The angle between the two tangents T_i and T_j must be as close to 180 degree as possible, that is, the split line needs to be discarded if the condition, $\left(T_i^{\,T} \cdot T_j\right) > \lambda_T$, is satisfied, where λ_T is a predefined threshold.

- The split line must be approximately perpendicular to the two tangents, that is, the values of β_{ij} and β_{ji} must be close to 90 degrees. In other words, a split line must be ruled out if the condition

$$\max\left(\left| T_i^{\,T} \cdot {e_{ij}}/{\left|e_{ij}\right|} \right|, \left| T_j^{\,T} \cdot {e_{ji}}/{\left|e_{ji}\right|} \right| \right) > \lambda_\beta,$$

  where λ_β is a predefined threshold, is satisfied.

Once the geometrical constraints are applied the final step is to get the final set θ^* of the split lines. All the split lines associated with the concavity points, which have a degree (number of split line through it) equal to one, are accepted in the output list. All the remaining split lines with concavity points that are already included in the output list are discarded. If there are split lines forming a triangle, then the pair of split lines that give minimum convexity measure is retained. Convexity is defined by

$$C = -N + \sum_{i=1}^{N} \frac{\emptyset_i}{\pi}, \tag{3.17}$$

where $\emptyset_i$ is the sum of angles of the tangents on the contour of i_{th} partition and N is the total number of such partitions after splitting.

After the execution of the above mentioned steps, the finally obtained output list, referred to as θ^*, is the list of pairs of concavity points which are joined by lines to segment the clumps into individual convex objects.

3.2 Mathematical Morphology-Based Methods

Mathematical morphology-based methods are quite simple and widely used methods for separating cells from background as well as from the other cells. Many of these methods are developed using basic morphological image processing, such as erosion, dilation, opening, and closing, sometimes combined with certain image segmentation algorithms as well. The literature has a lot of methods [7, 14, 20, 26, 34] which can be classified as morphology-based methods, on the basis of the approach used to develop them. However, there exists a problem using them in that they do not deliver accurate results when the cells cluster heavily, forming large cell clumps. In these cases, over or under-segmentation or both occur quite often.

Due to this problem, during this work, our sole purpose was to use the results of this widely used approach for comparison with our implemented method so as to emphasize the importance of our work. Here we discuss only the basic method from this approach which is not from a particular author, but in fact a more generalized approach using morphological image processing.

The first step involves the construction of the marker image for the watershed transform and there are variations in deriving the marker image. The purpose of the marker image is to control, though it does not completely remove, the oversegmentation inherent in the watershed algorithm. Here, we discuss three different approaches to generate the marker images.

In [20], the authors use a distance transform of the inverted binary image to convert it into a distance image. In the distance transform, the distance of every dark pixel from the nearest bright pixel is found using the Euclidean distance as the metric. This distance image is then opened by morphological opening, see Section 2.2.1, using a suitable structuring element to discard regions smaller than the expected cell size. The authors of [22, 32] apply *h*-maxima transformation to the distance transformed image to make sure that there is only one local maximum for every single object. The output is then inverted to create an image with one local minimum value for every individual object to be used as marker image for the subsequent application of watershed transform.

The authors of [14] propose the fusion of the distance transformed image with the multi-scale morphological gradient image to get the marker image. Since the morphological gradient, described in Section 2.2.3, relies on the size of the structuring element, the multi-scale approach is employed here. It uses both small and large structuring elements by

$$M(f) = \frac{1}{n}\sum_{i=1}^{n}\{[(f \oplus B_i) - (f \ominus B_i)] \ominus B_{i-1}\}, \quad (3.18)$$

where B_i is a structuring elements having size (2i+1) by (2i+1), f is the gray-level image and M is the multi-scale morphological gradient. This averaging also makes the measure more immune to noise than the simple gradient measure. After passing through the morphological opening procedure this gradient image is combined with the distance image to get the marker image for the watershed algorithm.

The use of morphological granulometry, described in Section 2.2.3, for finding the size distribution of the image objects is also a good technique to get the marker image. The authors in [26] proposed to find the size s from the size distribution, and use a disc shaped structuring element (considering round objects) of the same size to perform the morphological opening of the image. The resulting image after the application of the morphological gradient on it gives a good marker image for the subsequent application of the watershed algorithm.

Once the marker image is obtained, the watershed algorithm is applied on it, with the constraint that the markers are the only regional minima [13]. The obtained watershed lines are then used to define the cell contours of the final segmented image, which, if obtained accurately, contains every cell separated from the clump to which they belonged initially.

3.3 Model-Based or Parametric Fitting-Based Methods

More often than not, cells in microscopic images are elliptical and can easily be modeled with an ellipse with not much difference in the lengths of the major and minor axes. Thus some kind of template matching or ellipse fitting on the contour of the cell images can be effectively used to split cell clumps. There are numerous methods in the literature falling in this category [2, 6, 17, 30, 42]. Usually the methods from this approach are parameter-dependent. A complete review of the various methods is beyond the scope of this thesis. Hence we discuss a general approach used in these methods reviewing the methods presented in [2, 6].

Before fitting an ellipse to split cells from cell clumps, the initial step in these methods is polygon approximation of the contours of the cells and cell clusters. The purpose of polygon approximation is to smooth the contour of cells and cell clusters which may be affected by noise. This is needed because the next step is to find the concavity points along the contour and then fitting the ellipse on the contour which may be problematic if the contour is noisy.

The idea of polygon approximation is simple: two non-consecutive points on the contour are taken and a straight line is drawn between them. Next the distance of every contour point, between those two points, from that line is found. If any of the points has a distance value greater than a specified threshold, then a line is drawn between this point and the first of the initial two points. Moreover it is taken as the first point of the next two points, otherwise the initial line is retained and the second of the initially chosen points is taken as the first of the next two points until we reach the end of the contour points. During this process we keep discarding intermediate points, as we take only the end points to make the line, so that finally the number of points on the contour is less than originally.

After polygon approximation, the concavity points are needed because they are actually the points at which the contour is segmented into different small curves. There are many different techniques to find these points such as the curvature based method [39] and the tangent based method [17]. The idea is to take all those points which are located in regions that are concave. The concavity should be due to the merging of different individual cells rather than due to some irregularities or noise in the cell contour. From each of those regions, one or more points are taken which has the maximum value of concaveness in that region and those points are called concavity points.

Once concavity points are found, the cell contours are divided into contour sections from those concavity points. Naturally, these are the points from where the cells need to be segmented from the clump. In case of a cell clump, these contour sections are parts of the different cells. For one cell, there may be more than one contour section but one contour section belongs to only one cell.

Next, an ellipse is fitted on each contour section. There are many ellipse fitting techniques, least squares fitting [11] being the most popular one. Once an ellipse is fitted to every contour section then, there is some evaluation criteria applied to those ellipses. For example, the ratio of the major to the minor radii should not be larger than a specified threshold and the ellipse contour should contain maximum points from the contour section. Any ellipse that fails to get through any of these criteria is rejected.

It often happens that two or more contour sections of the same cell have an ellipse fitted to each of it after the ellipse fitting step. Therefore the next step is the combination of those ellipses so that both the contour sections share a single ellipse due to being part of the same cell as shown in Figure 3.6. It is sometimes referred to as grouping of contour sections and completing the boundary of the cell they are part of. There are certain aspects that are taken into account while deciding whether two ellipses are to be combined or not. One of them is the distance between the centers of the two ellipses; if it is a larger value than a specified threshold, then ellipses should be retained. Also the distance

from the center of the newly fitted ellipse to the center of previously fitted ellipses is taken into account in order to decide on combining the ellipses.

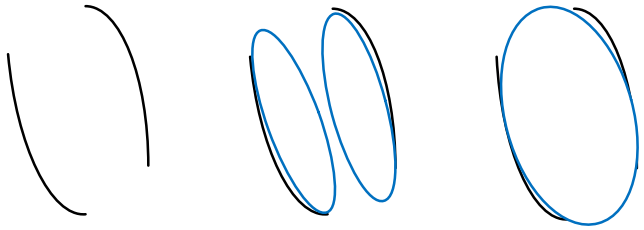


Figure 3.6: *Ellipse fitting and ellipse combination. (a) Two contour sections of a single cell with (b) two different ellipses fitted on it. (c) Combination of the two ellipses of (b).*

Finally, there can be some post processing performed on the ellipse fitted image. One example is finding of the medial curve for two overlapping ellipses so that the output image contains the original cell image with curves separating individual cells from the clump.

Chapter 4

New Methods for Clump Splitting

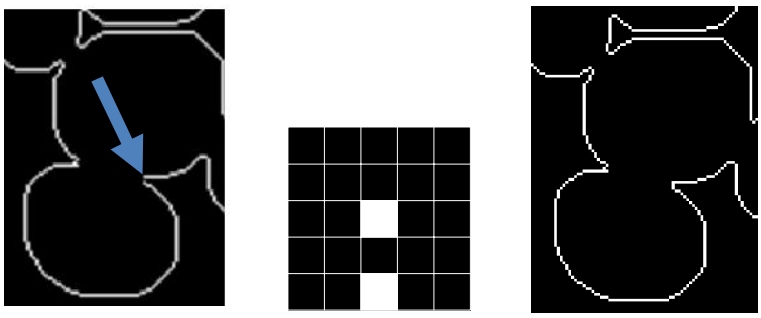
The approaches and the methods that we described in the previous chapter were found to result in false split lines when the clumps are more complex. However, there were some useful features in some methods, and if those features are collected and combined together with some modifications in the faulty segments, then we can obtain a significantly improved method. In this chapter, we first present a method, also presented in [8], that is obtained in a manner discussed above and the achieved results are close to the desired one. Then we move on to describe a novel method for clump splitting which is developed during the course of this thesis. Although the steps followed are similar to the previously studied methods, the techniques to perform them are novel.

In order to apply either of the clump splitting methods, the images are first passed through the image pre-processing steps; later on some post-processing is also necessary in order to get the split lines closely resembling the ones obtained manually. Before the application of the clump splitting methods it is assumed that the image is already segmented and transformed into binary.

4.1 Image Pre-Processing

In the preprocessing phase, the binary image is first zero padded. This is needed because we may have certain images in which objects touch the image boundaries and finding the contour of the closed shape objects is not possible in that case. The contour of the image objects is then found. There are several methods to do that but the effective ones are presented in [4, 5]. Then, coordinates of the points of the object contour are found for each object. If there are any discontinuities in the detected contours, they are removed. This is done by taking the 3x3 neighborhood of the current contour pixel (where the discontinuity is) and making the center element of the 3x3 block (i.e. the value of current pixel) as 0. If there is only one or no other bright pixel in this 3x3 block and also there are bright pixels in the 5x5 neighborhood of the current contour pixel, then there is

a discontinuity which needs to be removed. To do that, the 5x5 neighborhood of the current contour pixel is taken as the current block, and, depending on the location of bright pixel in the 5x5 block, one of the pixels of the 3x3 block is made to be bright, which is then placed into the original image. Labeling is then done to get the number of connected objects. Figure 4.1 illustrates an example case of removing discontinuities.



(a) (b) (c)

Figure 4.1: *Contour discontinuity removal. (a) Part of an image of size 160x120 pixels showing broken contour with an arrow. (b) The 5x5 window centered at the current pixel in (a) pointed by the arrow. (c) Image after the gap is filled by putting a "1" at the empty location between the two bright pixels.*

Next, every labeled connected object is taken and its contour pixels are traced in a clockwise direction, starting from the left-topmost pixel (see Figure 4.2 for illustration of the process). The purpose is to make a list of spatial coordinates of the contour pixels along with their index values. This is done by using an initial mask (see Figure 4.2(b)) which starts the search in a clockwise direction. Once started, it can take any of the 8 possible directions (see Figure 4.2(d)). This is because the mask (see Figure 4.2(e)) favors all of them. Therefore, a 3x3 distance matrix (see Figure 4.2(c)) is used to minimize the conflict between the possible directions. If there is more than one direction to follow then in the case of conflict between horizontal and vertical locations with respect to current contour pixel (see Figure 4.2(f)) the 5x5 neighborhood of the current contour pixel (see Figure 4.2(h)) is used to select the next contour pixel. In the other case of conflict between diagonal locations with respect to current contour pixel (see Figure 4.2(g)) the previous pattern is used to decide where to go next by using the 3x3 window and including the previous pixel (see Figure 4.2(i)).

While moving to the next contour pixel, a 0 is placed at the previous contour pixel in the original image so that search proceeds in the forward direction (otherwise it could return back to the previous location). Also, before going to search every next contour pixel location, it is checked if the 3x3 neighborhood of the next contour pixel contains more than one bright pixel, otherwise, it is the last pixel of the object or there was a problem in the contour found initially. In the former case, the coordinate values of the contour pixels of the currently analyzed object of the image are returned, and in the lat-

ter case, that contour pixel is removed from the image and the whole process is repeated again. The whole process is depicted in Figure 4.2. Once all the objects of an image are processed, then those individual object images are added to make the pre-processed version of the original image of contours.

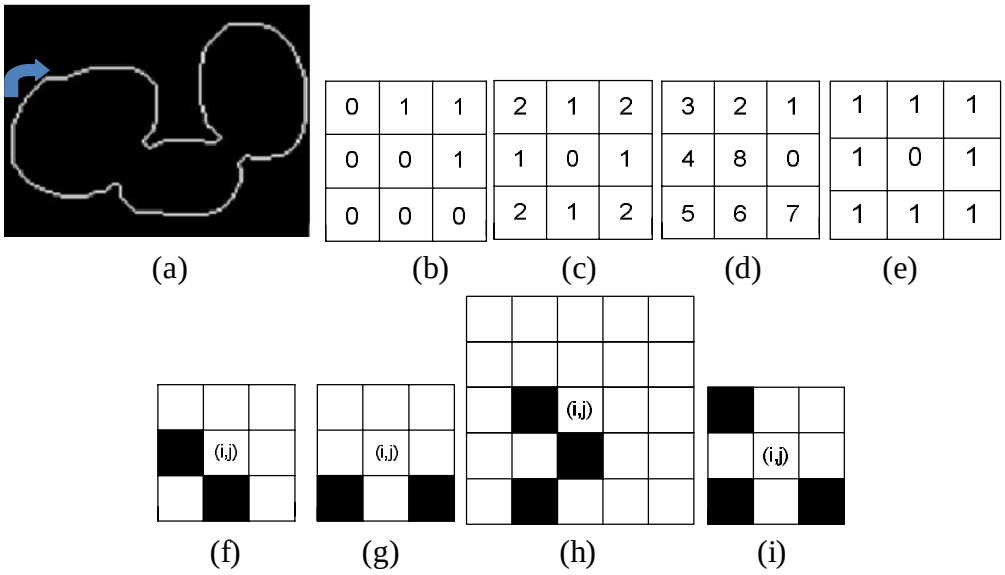


Figure 4.2: *Clockwise contour tracing. (a) Clockwise tracing of contour of an object in an image of size 180x230 pixels. (b) The initial 3x3 mask to start searching for the next pixel in clockwise direction. (c) The distance values for next positions. (d) The labeled possible directions. (e) 3x3 mask for searching 2^{nd} pixel and onwards. (f) and (g) are possibilities available at one time which are solved by (h) and (i) respectively.*

4.2 Modified Clump Splitting Method

The method presented in this section, also presented in [8], is derived from Kumar et al. [18]. The key features of this method are used with some changes in them, in order to make the method less prone to error. Along with those features, two key features from the method in Wen et al. [39] are also included in this method to get the improvement in the relevant aspects of the overall clump splitting method.

The method for clump splitting starts with finding the concavity points in the image. Then Delaunay triangulation is applied to obtain the candidate split lines from which the best split lines are obtained for every valid concavity point such that a cost function is minimized for that particular concavity point.

4.2.1 Detection of Concavity Points

All the concavity points on the boundary of the objects are first found. In all the available methods, the best approach for the determination of concavity points is using curvature-analysis, defined by Equation 3.17. It finds all the concavity points in a particular concavity region with the curvature values larger than a predefined threshold. This is an improvement in comparison to the approach used in [18] which defines only one concavity point per concavity region. Figure 4.3 shows a clump of concave objects in which there is more than one concavity point in a concavity region. In the concavity point based analysis, it is obviously important to locate all the valid concavity points in the initial phase. Here we used the methods from [15, 16] in which the corners in the image are detected using curvature analysis. The detected corners are examined such that only the points lying on the concave regions of the contour are picked. This gives almost all the valid concavity points in the image.

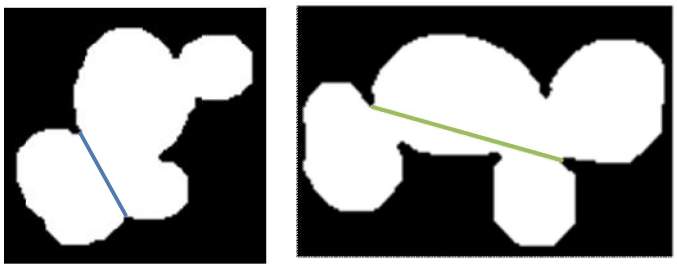
Figure 4.3: *Concavity regions with multiple concavity points. A Clump of cells with more than one concavity point in one concavity region, marked with white squares.*

4.2.2 Listing Candidate Split Lines

At this point we have N concavity points and we can have $\binom{N}{2}$ split lines joining these concavity points. Obviously not every line joining two concavity points can be a split line. We therefore apply Delaunay triangulation, introduced in [39], on the found concavity points to exclude invalid split lines. The remaining possible split lines are then referred to as the candidate split lines. The advantages of Delaunay triangulation are that it rules out the chance of intersecting split lines as well as maximizes the minimum of all the angles of the triangles that are made by the split lines. These two points are much needed since we do not want the split lines to intersect each other nor do we want small angles between the final split lines.

4.2.3 Finding the Best Split Lines

Although most of the invalid split lines are discarded by the application of Delaunay triangulation on the set of concavity points, some invalid split lines still remain, and they are removed by applying a set of conditions on the features that are extracted from the image for every concavity point. The features that we used here are taken from [18] but they are modified in order to remove the deficiencies present in them.



(a) (b)

Figure 4.4: *Saliency problem. In (a), due to small concavity depth the valid blue split line makes the saliency value small enough that the green line in (b),in one of objects of the same image set, becomes possible. Since the value of alignment also supports that line it is possible that this green line is actually made.*

4.2.3.1 Saliency

The first feature that is used here is the saliency feature, which takes into account the concavity depth [25] of the concavity points as well as the distance between them, as given by Equation 3.11. The idea behind the saliency feature is that the split lines are more likely to be valid if the concavity regions at both ends of the line have large enough concaveness measures and the distance between the two concavity points is small [18]. However, the problem in that expression of saliency is that, as the value of the minimum concavity depth increases, then for a particular saliency value the allowed distance between the two concavity points also increases [8]. This makes it very difficult to find a threshold value that result in a correct split line in all the cases. That is, if one of the two concavity depth values is small but a split line is valid, then even a small value of the distance between the two concavity points makes the saliency small. But due to non-linear increase in distance with respect to the decreasing saliency values, putting a very small saliency threshold would lead to the acceptance of long distance lines in the cases where the minimum concavity depth values are large. Figure 4.4 depicts these situations.

In order to support our above arguments about incorrect definition of saliency expression in Equation 3.11, we construct a table (see Table 4.1) which highlights the nonlinear relationship between saliency, minimum concavity depth and the length of a split line. For example, Table 4.1 shows the situation where for a minimum concavity depth

value of 10 pixels the value of saliency for a 25 pixel line is such that it would allow lines of length 35 pixels and 50 pixels for minimum concavity depth values of 15 pixels and 20 pixels, respectively. Also this non-linearity increases with distance values. Thus the main thing that we need is to define an expression of saliency such that it minimizes the non-linear decrease in the saliency values when both the minimum concavity depth and the distance values increase. We tried a few different expressions; one of them was to include both of the concavity depth values, but Table 4.2 show that this does not produce the desirable results.

Table 4.1: *Values of Saliency SA obtained using the original expression, in Equation 3.11, for different distance and concavity depth values.*

Distance Values	SA	SA	SA	SA	SA
	CD(min) = 10	CD(min) = 15	CD(min) = 20	CD(min) = 25	CD(min) = 30
10	0.500	0.600	0.667	0.714	0.750
15	0.400	0.500	0.571	0.625	0.667
20	0.333	0.428	0.500	0.556	0.600
25	0.286	0.375	0.444	0.500	0.545
30	0.250	0.333	0.400	0.454	0.500
35	0.222	0.300	0.364	0.417	0.461
40	0.200	0.273	0.333	0.385	0.429
45	0.182	0.250	0.308	0.357	0.400
50	0.167	0.231	0.286	0.333	0.375
55	0.154	0.214	0.267	0.312	0.353
60	0.143	0.200	0.250	0.294	0.333
65	0.133	0.187	0.235	0.278	0.316
70	0.125	0.176	0.222	0.263	0.300

We therefore came up with a modified expression of saliency which takes into account the increase in both the minimum concavity depth values and the distance values by scaling them accordingly. That is, using large coefficients for scaling large values and small coefficients for scaling small values. This is achieved by using the squared values of both of the parameters in the denominator, thus the modified expression for saliency is

$$SA_{i,j} = \frac{\min(CD_{i,j})}{0.1*\min(CD_{i,j})^2 + D_e(C_i,C_j)^2}, \quad (4.1)$$

Table 4.2: *Values of Saliency SA for different distance and concavity depth values obtained by using* $SA_{i,j} = \frac{CD_i + CD_j}{CD_i + CD_j + 2 * D_e(C_i, C_j)}$.

Distance Values	SA	SA	SA	SA	SA
	$CD_i + CD_j$ = 20	$CD_i + CD_j$ = 25	$CD_i + CD_j$ = 35	$CD_i + CD_j$ = 50	$CD_i + CD_j$ = 60
10	0.500	0.556	0.636	0.714	0.750
15	0.400	0.454	0.538	0.625	0.667
20	0.333	0.385	0.467	0.556	0.600
25	0.286	0.333	0.412	0.500	0.545
30	0.250	0.294	0.368	0.454	0.500
35	0.222	0.263	0.333	0.417	0.461
40	0.200	0.238	0.304	0.385	0.429
45	0.182	0.217	0.280	0.357	0.400
50	0.167	0.200	0.259	0.333	0.375
55	0.154	0.185	0.241	0.312	0.353
60	0.143	0.172	0.226	0.294	0.333
65	0.133	0.161	0.212	0.278	0.316
70	0.125	0.151	0.200	0.263	0.300

Table 4.3: *Values of Saliency SA obtained using the finally obtained expression, in Equation 4.1, for different distance and concavity depth values.*

Distance Values	SA	SA	SA	SA	SA
	CD(min) = 10	CD(min) = 15	CD(min) = 20	CD(min) = 25	CD(min) = 30
10	0.0909	0.1224	0.1429	0.1538	0.1579
15	0.0426	0.0606	0.0755	0.0870	0.0952
20	0.0244	0.0355	0.0455	0.0541	0.0612
25	0.0157	0.0232	0.0301	0.0364	0.0420
30	0.0110	0.0163	0.0213	0.0260	0.0303
35	0.0081	0.0120	0.0158	0.0194	0.0228
40	0.0062	0.0092	0.0122	0.0150	0.0178
45	0.0049	0.0073	0.0097	0.0120	0.0142
50	0.0040	0.0059	0.0079	0.0098	0.0116
55	0.0033	0.0049	0.0065	0.0081	0.0096
60	0.0028	0.0041	0.0055	0.0068	0.0081
65	0.0024	0.0035	0.0047	0.0058	0.0070
70	0.0020	0.0030	0.0040	0.0050	0.0060

where, as previously $CD_{i,j}$ refers to the concavity depths of the two concavity points and $D_e(C_i, C_j)$ is the Euclidean distance measure. The obtained values for saliency are listed in Table 4.3, which shows that the current expression addresses the aforementioned problem of Equation 3.11. A large value of $SA_{i,j} > 0$, practically between 0 and 1, is required to ensure the candidacy. Next, for finding the best split lines, the saliency value is thresholded so that large distance lines are taken away from further calculations.

4.2.3.2 Alignment

The second feature that is employed here to narrow down the list of candidate split lines is the alignment feature. It contains concavity-concavity alignment (CC) and concavity-line alignment (CL) features, as defined by Equations 3.12 and 3.13, respectively. The alignment features heavily rely on the directional vectors of the concavity points involved, and a slight error in the calculation of that parameter leads to false split lines. Thus the problem may arise when the directional vector, v_i, is computed for a concavity point, C_i.

The definition of directional vector that was given in [18] is that it is a vector of length unity originating from the midpoint of the corresponding convex hull chord with its head towards the concavity point. For a concavity region which has only one concavity point, a good directional vector is often obtained according to this definition. However, this is not the case with every concavity point, since there are concavity regions which have more than one concavity point. In that case, taking the midpoint of convex hull chord as the other point of the directional vector leads to inappropriate directional vector. Moreover, sometimes the shape of the concavity regions is such that even if it has only one concavity point, the obtained directional vector does not satisfactorily achieve the purpose that it is used for. Hence, if the earlier definition of directional vector of [18] is used the obtained values of the alignment features are not always correct. In particular, in cases where there is more than one concavity point in a concavity region the directional vectors obtained by the previous definition do not conform to the natural condition that the directional vector must approximately bisect the region close to the concavity point rather than trying to bisect it from the convex hull chord. Figure 4.5(b) shows some examples in which it is clear that the found directional vectors are not perfect. Instead the directional vector should be a line which would split the region around the concavity point as illustrated in Figure 4.5(c). Another example is shown in Figure 4.5(d) where the red vectors found with initial approach must be replaced with the blue ones found by our method.

We developed a new approach for finding the correct directional vectors. As we already mentioned in Section 4.1 that while moving on the contour of a clump, we made a linked list of spatial coordinates of the contour with their index values. With that, for a

particular coordinate value its index can be obtained and hence the neighboring contour points can be accessed through incrementing or decrementing the index values.

In order to find the directional vector of a concavity point its index value is obtained first. Using that index value two points on the contour, one on either side of the concavity point, are accessed. The midpoint of the straight line connecting these two points is found. This point is then used as the tail-point of the directional vector. The head of the directional vector of course lies in the line connecting this point to the concavity point.

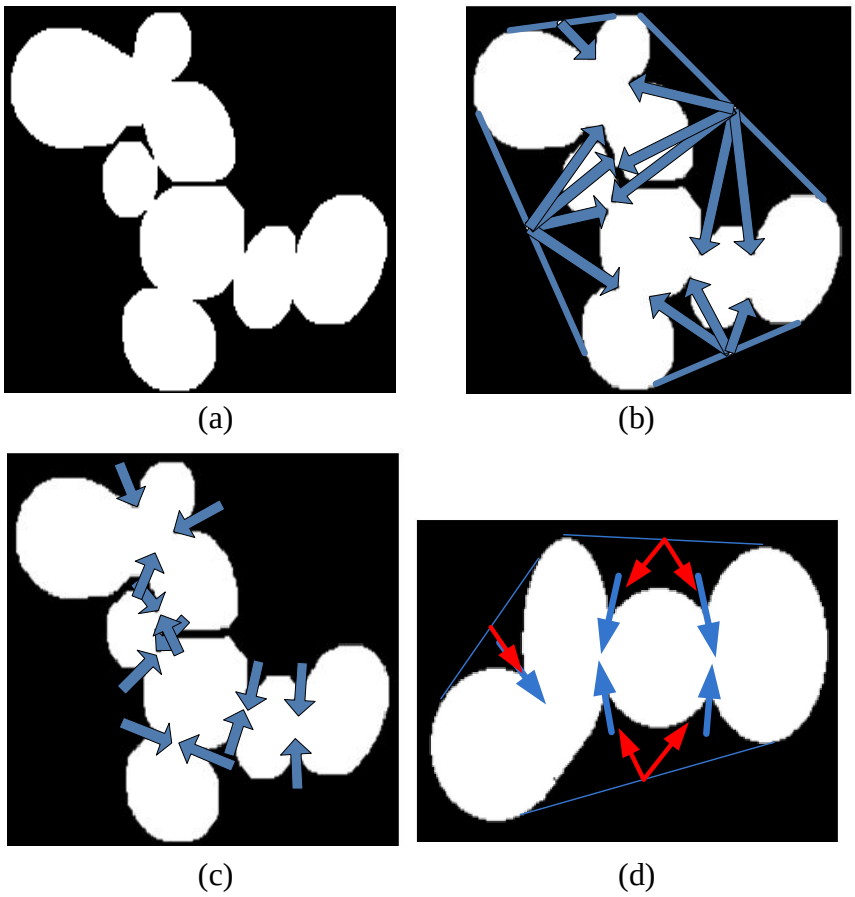


Figure 4.5: *Directional vector problem. (a) A Clump of cells. (b) Incorrect directional vectors for concavity points found by the definition in [18]. (c) Correct directional vectors found by the modified method. (d) Another such clump with inaccurate directional vectors in red and correct ones in blue.*

4.2.3.3 Cost Function

Once these features have been calculated for every possible concavity point-pair, they are utilized to evaluate a cost function for each point-pair. We assume that every valid concavity point is the consequence of two cells in contact with each other and that there must be a split line for every concavity point. With this assumption, the best split line is defined for every concavity point based on a cost function that is evaluated for every

possible pair of each concavity point. The point which gives the minimum value for the cost function is selected as the pair for that concavity point. The reason why the minimum value is needed is that every feature contributing in the cost function needs to be minimal for the best split line. Thus the idea is to pick the best split line for every concavity point even if it is duplicated but in the meantime reject those concavity point pairs where both the points have already been used in another split line. This way all the correct and valid split lines are taken into account and also the false split lines are ruled out.

The expression for the cost function that we used is obtained by summing the saliency, CC alignment, and CL alignment feature multiplied by two and is given by,

$$CF_{i,j} = SA_{i,j} + CC_{i,j} + 2 * CL_{i,j}. \quad (4.2)$$

The logic behind multiplying the CL alignment feature by two is that sometimes two points are not well aligned with each other, but due to their directional vectors being almost perfectly anti-parallel, they get selected instead of those pairs for which the split lines are more aligned to both directional vectors, see Figure 4.6 for an example. Also, the range of the values for CC alignment is from -1 to +1 whereas for the CL alignment it is from 0 to 1, so the CL feature needs to be doubled in order to treat both features equally, and also to have a better look into the pairs before deciding about the best split lines. This fact is demonstrated in Figure 4.6 where we select the split line which is less anti-parallel and a little more aligned with the directional vectors.

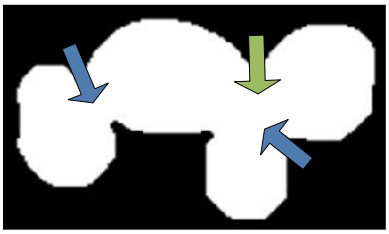


Figure 4.6: *Alignment problem. Concavity point-pair marked by two blue arrows has a higher value of CC alignment than the pair marked by green and blue arrows in the right. However the latter pair gives correct split line when both the alignment features are examined simultaneously.*

There may be a case that a clump has only one concavity point or a concavity point is left without a pair after the split lines are obtained. In such a case, a split line is found for that concavity point by joining it with a boundary point of the object in the direction of its directional vector in the same way as it was described in Section 3.1.3. After all the split lines are found, they are drawn on the binary image. If the split lines result in a

very small object then that object is removed. The resulting image is clump splitted version of the original image.

4.3 Clump Splitting Method using Variable Size Rectangular Window-Based Concavity Point-Pair Search

The clump splitting method presented in this section is also a concavity point-based method in which the process starts with finding the concavity points on the contours of the objects. This is followed by finding the split lines which connect these concavity points, provided that certain conditions are fulfilled. There are a few concavity point detection methods which provide satisfactory results. However, the main problem that is faced in clump splitting is the selection of the best concavity point-pairs for the split lines. The available methods for this do not perform up to the expectations and result in unsatisfactory clump splitting, as false split lines are often obtained or true split lines are rejected. Although the results that we achieved with the modified method are better than the results obtained with the methods available in the literature, the problem with that method is that it relies on considering more than one concavity point while looking for the pair for a particular concavity point. Due to this reason we need to take into account different parameters in order to obtain the best split line. Since we often have a large set of images containing different kinds of clumped objects it can be difficult to obtain a set of parameters which is applicable to all the objects in every image in a particular image set.

To solve this problem of finding the best split lines and to make the algorithm less dependent on the parameter values, we propose a new method for finding the best split lines. It uses a variable size rectangular window to search for the pair of a particular concavity point. Similar to what we proposed in the modified method described in Section 4.2, this method also finds the best split line for every concavity point, whether by joining it with other concavity point or with a boundary point on the other side of the concavity point.

4.3.1 Detecting Concavity Points

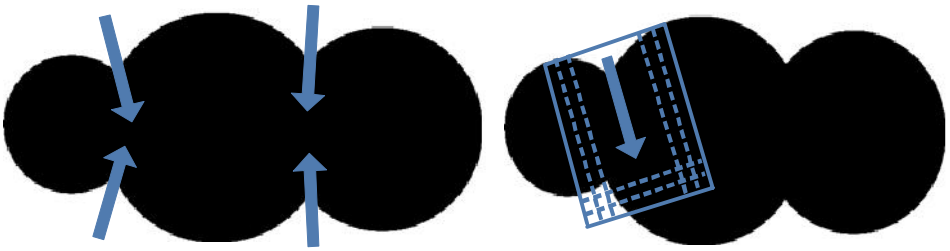
The first step is to extract all the concavity points that are formed due to two objects being in contact with each other. Since it is assumed that every obtained concavity point is valid, concavity point detection needs to be done carefully such that no single point is taken which is there just because of boundary irregularities.

There are several methods [10, 37, 43] in the literature which detect the concavity points in clumped objects along with the one that we used quite effectively in the previous method [39]. Although that method gives much better result than others, there is still some room for improvement in it.

To make sure that the concavity point is valid we find the concavity angle, i.e., the angle between the lines joining the concavity point with equally distant points on either side of the concavity point on the object contour. For a concavity point to be valid the concavity angle should be as small as possible. Thus we discard points due to boundary irregularities which give angle values larger than 150 degrees. Moreover, there are some cases in which we have both valid and invalid concavity points having the same appearance. This is problematic, because if we take both of them then we may get over-splitting. However, if we exclude both of them then it may not be costly because in that case there is a chance that the likely pair would make a split line joining a boundary point on the opposite side of the contour resulting in the final output being not much affected.

4.3.2 Searching for the Best Split Lines

In Section 4.2.3, it was described that there are cases in which it is very difficult to decide which split line is the best one. This is because the values of the alignment features leave it uncertain which split line should be picked. Although this issue is dealt with by defining the cost function in such a way that it weighs the different features differently, however sometimes false split lines get selected. Thus, in order to focus on finding only the best split lines, we take into consideration the fact that the split lines should be within a specific region in the direction of the directional vector of the concavity point. The directional vector is defined in the same way as in Section 4.2.3.2.

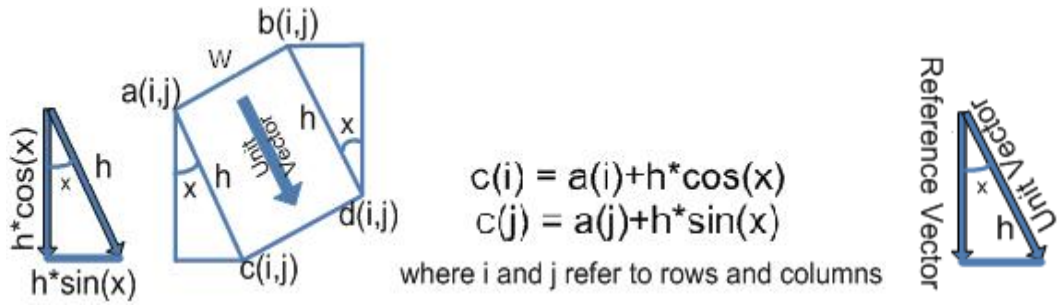


(a) (b)

Figure 4.7: *Rectangular window-based concavity point-pair search. (a) Clumped object and the orientation of directional vectors of concavity points. (b) Rectangular window with varying width and length (shown with dashed lines) aligned in the same direction as the directional vector in order to search for the concavity point-pair.*

We mentioned in Section 4.2.3.2 that the directional vector that defines the alignment of a concavity should ideally bisect the region close to the concavity point. This idea leads us to start searching for the concavity point pair in the direction of the directional vector and on either side of the line formed by extending the directional vector. This gives rise to a variable size rectangular window in the direction of the directional vector. This concept is illustrated in Figure 4.7.

The idea basically is that we start from the concavity point and find its directional vector. Then by traversing along the contour of the object we pick one point on either side of the concavity point and both equally distant to it, such as points *a* and *b* in Figure 4.8(a). The distance between these two points is referred to as width of the window, *w*. These two points are then used as two of the four corner points of the rectangular window. The other two points, points *c* and *d* in Figure 4.8(a), are found in the direction of the directional vector and at equal distance to *a* and *b*, respectively. This distance value, *h*, is a parameter which defines the length of the window and depends on the maximum length of the split line that we want to allow in a certain image set. Now, in order to find those two points we first need to know the angle, *x*, between the directional vector and a reference vector, see Figure 4.8(b). This angle is used to calculate the horizontal and vertical displacement from the current points by using the simple trigonometric relations on the triangle formed, as shown in Figure 4.8(a).



(a) (b)

Figure 4.8: *Finding the co-ordinates of the window. (a) Formation of the window by using the already found points a(i,j) and b(i,j), directional unit vector and the reference vector to get the points c(i,j) and d(i,j). (b) Vector diagram showing the directional unit vector and reference vector and the angle formed between them.*

Once we have found the displacement values, we then need to add or subtract these displacement values to the spatial coordinates of the current points, that is, points *a* and *b*, in order to reach the desired points *c* and *d*. The directional vector is used to decide whether addition or subtraction is to be performed. It is basically the sign of the *x*- and *y*- components of the directional vector which determines whether the displacement values are added or subtracted. Negative sign indicates that the particular component is to be subtracted from the respective component of the spatial coordinate of the current

point. Thus, on the basis of the direction of the directional vector there are four general orientations of the window as shown in Figure 4.9.

It is shown in Figure 4.9 that in order to get the correct angle values, we need to have two reference vectors, but this can also be done with only one reference vector. In the case of opposite orientation, the value of the obtained angle would be larger than 90 degrees. Thus finding the angle between the directional vector and that reference vector and subtracting the angle from 180 degrees gives us the right value of the angle.

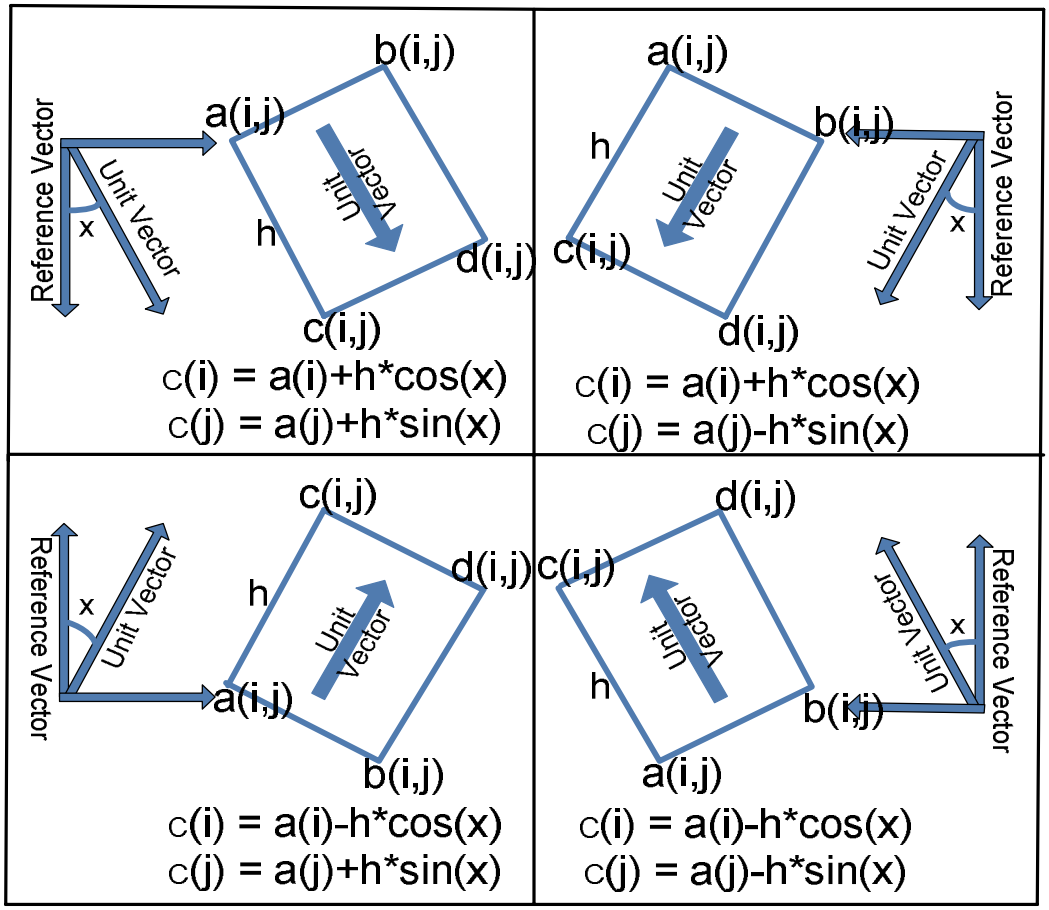


Figure 4.9: *Possible orientations of the window. Four different orientations of the window based on the direction of the directional unit vector and the respective reference vectors and co-ordinate calculations.*

Once we get the coordinates of the four points of the window, we use this window as the mask for finding the concavity point pair to make the split line. Initially we set minimum width and a considerably large length of the window and look if there is any concavity point lying in this window. We then gradually increase the width of the window until we find a concavity point inside it or reach the maximum width. Starting with a small window width prevents getting two concavity points in the search window. In the

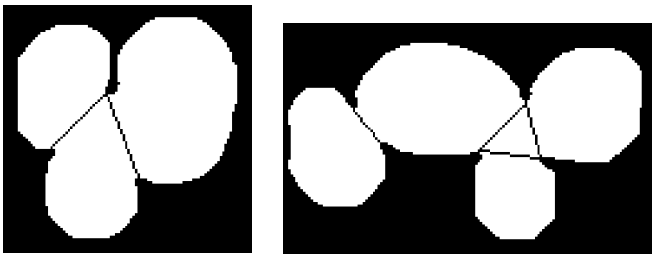
case that there are two concavity points in the search window, the concavity point at minimum length from the subject concavity point is accepted as the pair. If no concavity point is found when the maximum window width is reached, the window length is increased iteratively until a concavity point is found or the maximum window length is reached.

This whole process is repeated for every concavity point present in the object and a list of the concavity point-pairs is formed. This list is then checked to remove the redundancy and then all the split lines are drawn on the object by joining every concavity point-pair. Once we are done with this then we look for any concavity point which was not assigned a pair earlier. In most cases the pairing concavity point for such a concavity point was discarded in the initial concavity point detection phase due to lack in concaveness or due to the presence of irregular boundaries.

In such cases and in the case when there is only one concavity point in a clump (see, for example, Figure 3.4(c)), we draw a line from that concavity point to a point on the boundary of the object in the direction of the directional vector of the concavity point. The boundary point may not necessarily be the point of intersection of the object contour and the line formed by the extension of directional vector. In fact, it is chosen from a contour segment such that it possesses a local maximum value of concaveness among a certain number of pixels in that segment of contour.

4.4 Image Post-Processing

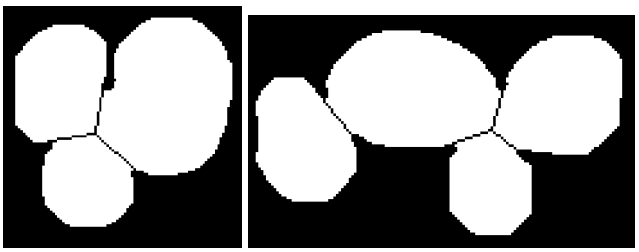
The clump splitting methods described in the thesis merely define straight lines between the concavity point pairs and the relationship between the split lines is not taken into account. For example, sometimes we confront a situation in which there are two split lines through a particular concavity point making an acute angle between them, as shown in Figure 4.10(a). Moreover, sometimes the other two concavity points involved in the two split lines also share a split line between them which forms a triangular object in an image, as illustrated in Figure 4.10(b). If there is prior information that the objects have smooth boundaries, both of these cases lead to an output image which does not look pleasant and also in the latter case we have an extra object which has nothing to do with the cells present in the image. Therefore, we need to post-process the clump splitted output image to make the final output look better and smoother.



(a) (b)

Figure 4.10: *Post-processing cases. (a) Object with two split lines making acute angle between them. (b) A triangle is formed between the three concavity points.*

These two cases can be solved in the post-processing as follows. We begin with finding the two cases by checking the degree of all the concavity points present in the object. The term degree is used to specify the number of split lines passing through a concavity point. So going through the list of concavity point-pairs we find out which of the concavity points have degree equal to two. Then we take the two other concavity points which share the split lines with the first concavity point and examine if they are forming a split line between them or they do not have any split line with any other concavity point. Once we get either of these conditions fulfilled then we form a triangle between the three points, if it was not already there, and find the centroid of that triangle. After finding the centroid we remove the concavity point-pairs formed by these three concavity points from the initial list and replace them with three concavity point-pairs each involving the centroid and one of the three concavity points. Figure 4.11 shows the output of the post-processing step for our example cases.



(a) (b)

Figure 4.11: *Result after post-.processing. The objects of Figure 4.10(a) and Figure 4.10(b) after the application of post-processing.*

Chapter 5

Results

In this chapter, we present results of clump splitting obtained using two methods from the literature, that is, the method from [18] discussed in Chapter 3 and the classic watershed-based method, as well as the two methods proposed in Chapter 4. The watershed-based method operates on the distance transformed binary image, see for example the description in Section 3.2 or [32]. The motivation for choosing this method is that it is probably the most widely used method for clump splitting despite its tendency to sometimes perform over- and under-segmentation. We gathered the results by applying these methods on two different test cases comprising of three image sets. The image sets contain images of budding yeast cells of different sizes and shapes. Moreover, we also present a quantitative comparison of these results in order to validate the proposed methods.

5.1 Test Case I:

For the first test case, we have two image sets containing bright field images of the budding yeast *Saccharomyces cerevisiae* obtained with a Leica TCS SP2 microscope. For every image in the set, z-stacks of 20 images were taken using a 100X oil immersion objective (NA 1.40). One of the 20 images was selected for segmentation by first finding the best in-focus image using the Tenegrad method [33], according to which, the z-slice that has the greatest normalized variance can be thought to be the most in-focus slice. Then the image that was one slice (about 300 nm) below the best focused image was selected, because more accurate segmentation results were obtained for this slightly out-of-focus image.

Both image sets contain cell clumps of approximately circular and elliptical cells. The difference between the two sets is that cells in the first set are bigger in size than the cells in the second set due to higher magnification used in imaging. However, the thing that is common between them is that most of the clumps that are formed by growing of bud from the mother cells contain cells of almost similar size for both the mother and

the grown cells. The cells are separated from the background using the segmentation method from [21].

For the validation process we use precision and recall analysis. We start from measuring precision (PR) and recall (RC) values from the clump splitted images by,

$$PR = \frac{TP}{TP+FP}, \quad RC = \frac{TP}{TP+FN}, \tag{5.1}$$

where TP, FP and FN stand for true positives, false positives and false negatives, respectively. Perfect precision means that all the single objects detected by the method are also present in the ground truth image, whereas perfect recall means that none of the objects in the ground truth image are missed by the method.

A more compact representation of the segmentation accuracy is obtained by using the F-measure (FM) [9]. The F-measure is the harmonic mean of the precision and recall measures and is given by

$$FM = \frac{2}{{}^{1}/_{PR} + {}^{1}/_{RC}}. \tag{5.2}$$

Through manual detection, we found that there are 772 single cells in Set 1 and 820 single cells in Set 2. We evaluated the performance of the aforementioned methods on the two test sets. We obtained TP, FP, and FN counts by manually going through the results of these methods and obtained PR, RC and FM values; see Table 1 and Table 2.

Table 1. *Performance values of the four methods on Set 1.*

	Total	TP	FP	FN	PR	RC	FM
Kumar Method	772	606	0	166	1.000	0.785	0.880
Watershed Method	772	747	24	25	0.969	0.968	0.968
Modified Method	772	733	13	39	0.983	0.950	0.966
Window Method	772	730	3	42	0.996	0.946	0.970

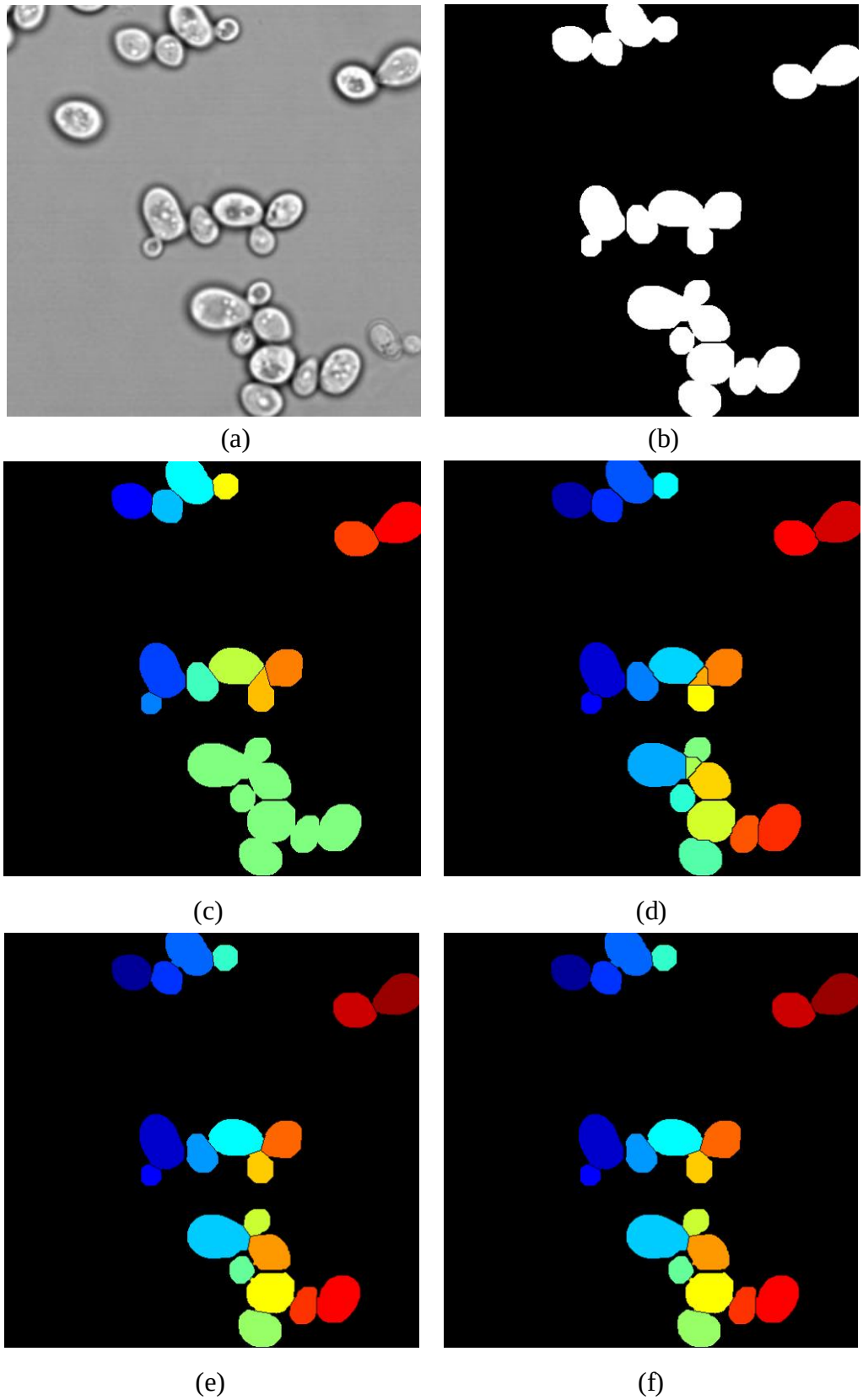


Figure 5.1: *Clump splitting results for test case I. (a) A bright field image of yeast cells. (b) Segmented image showing only clumps. The resulting image after application of (c) Kumar method, (d) watershed-based method, (e) modified method, and (f) window-based method on the image in (b).*

Table 2. *Performance values of the four methods on Set 2.*

	Total	TP	FP	FN	PR	RC	FM
Kumar Method	820	719	0	101	1.000	0.877	0.934
Watershed Method	820	810	16	10	0.981	0.988	0.984
Modified Method	820	788	0	32	1.000	0.961	0.981
Window Method	820	788	0	32	1.000	0.961	0.981

It is clear from Table 1 and Table 2 that both the modified method and the window-based method are superior to Kumar method and are comparable to Watershed-based method on the basis of the F-measure. Although in set 1 the precision for modified method and window-based method is slightly smaller than the precision for the Kumar method, both methods have a considerable gain over the Kumar method in the recall value. In both image sets, it is also visible that there is a trade-off between precision and recall measures for both of our methods and the watershed-based method. Of course, recall value for Watershed-based method is slightly higher than recall value for the modified method and the window-based method, but if we consider which method is making fewer false split lines then we would give more significance to precision values which is better for the two methods than for the watershed-based method.

Figure 5.1(a) is taken from the first image set and the resulting images illustrate the superiority of our methods over the Kumar method and the watershed-based method. It is clear from Figure 5.1 that the Kumar method under-splits some clumps because of taking only one concavity point in one concavity region and also due to the wrong definition of directional vectors. This is in line with the FN measure in Table 1, which is very high for Kumar method. On the other hand, the Watershed-based method over-splits some clumps which is noticeable by the triangular non-cellular objects that are detected as cells. This is also clear from Table 1 where watershed-based method has a high value for FP showing that sometimes it detects false positives. In contrast, the modified method as well as the window-based method split all the clumps correctly.

5.2 Test Case II:

For test case II, we have one set of images of yeast cells triply stained with FITC-ConA, taken from the Saccharomyces Cerevisiae Morphological Database (SCMD) [28]. The cells in this set of images are small and mostly the clumps are due to growing of buds from the mother cells where the size of bud is much smaller than the size of its mother cells. We took a random sample of 35 images containing 1080 yeast cells that belonged

to cell clumps. The images were segmented by a local thresholding method that is based on the classic threshold selection method by Otsu [23]. Details on the segmentation method can be found in [8].

We evaluated the performance of the four clump splitting methods on the segmented images. We obtained true positive (TP), false positive (FP), and false negative (FN) counts by manually going through the results of these methods and obtained precision (PR), recall (RC) and F-measure (FM) values, see Table 3.

Table 3. *Performance parameters of the four methods on set 3.*

	TP	FP	FN	PR	RC	FM
Kumar Method	807	11	273	0.987	0.747	0.850
Watershed Method	859	1	221	0.999	0.795	0.886
Modified Method	986	21	94	0.979	0.913	0.945
Window Method	982	20	98	0.980	0.910	0.943

Both of the new methods presented in this thesis have a significantly higher FM value than the Kumar method and the watershed-based method. The PR value for our methods is slightly lower than for the Kumar method and the watershed-based method, but their RC values are much higher.

Figure 5.2(a) is one of the 35 images constituting image set 3. Figure 5.2 (b)-(e) demonstrates the performance of the methods on the segmented image. It is evident that the two new methods presented in this thesis outperform the Kumar method and the watershed-based method. The Kumar method and the watershed-based method perform under-splitting, which is also obvious from looking at the FN values in Table 3. The modified method and the window-based method split the clumps more accurately which is also apparent from Table 3 where they have a lower value of FN as compared with the Kumar method and the watershed-based method.

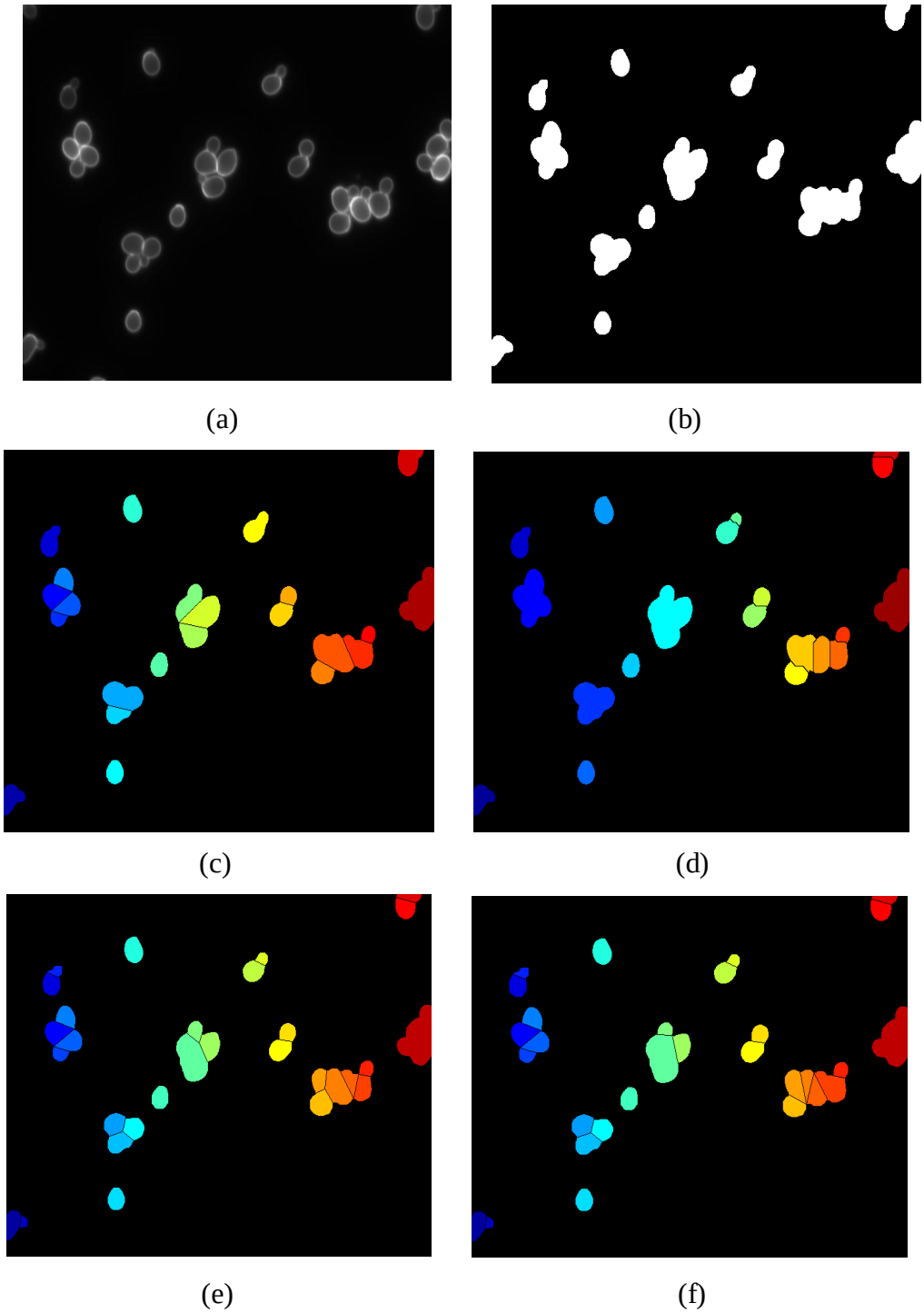


Figure 5.2: *Clump splitting results for test case II. (a) A fluorescent image of yeast cells. (b) Binarized image. The Resulting image after application of (c) Kumar method, (d) watershed-based method, (e) modified method, and (f) window-based method on the image in (b).*

Chapter 6

Conclusion

Two different methods for automated splitting of clumps of concave objects were proposed in this thesis. Both of the methods are based on concavity point analysis in which split lines are obtained by joining concavity points that are found on the boundaries of the clumped objects. The methods were tested on images of yeast cells but they can be applied in splitting clumps of any other convex object.

The first method proposed in the thesis, also presented in [8], is a modified version of the original method presented by Kumar *et al.* in [18]. Contrary to the original method, this method finds all concavity points in each concavity region of a clump. Also, this method makes sure that for a particular value of saliency the length of the split line increases approximately linearly when concavity depth increases. Moreover this method emphasizes the fact that the split line must split the clump from the vicinity of the concavity point rather than doing it from the mid-point of the corresponding convex hull chord.

The second method is novel and was developed during the course of the thesis. It uses a variable size rectangular window to look for the pair of each concavity point. One of the main advantages of this method is that it is less dependent on user-defined parameters than most other methods presented in the literature. This method takes advantage of the fact that for most of the split lines the pair of the current concavity point is found in the direction of the directional vector of the corresponding concavity. The method starts with a minimum window width in order to avoid the possible situation of having more than one concavity point in the window as the candidate pair of the current concavity point, and iteratively increases the window size until a pair is found.

The performance of both of the clump splitting methods was evaluated using two different test cases comprising of three test image sets altogether. All of the three image sets contain images of yeast cells. In order to validate the two methods, we compared them with the original method from Kumar *et al.* [18] and with the classic watershed-based

method, see for example [32]. Quantitative as well as qualitative comparisons demonstrate the superiority of our two methods over the two clump splitting methods taken from the literature. It was observed that the modified method outperforms the original method in both test cases. In fact, an increase up to 10 percentage points in F-measure, obtained using the harmonic mean of precision and recall measures, is achieved with the modified method. However, it was found to be comparable to the watershed-based method when the clumped cells are bigger and round shaped. On the other hand, the modified method performed better in the case when most of the clumped cells were small buds grown from the mother cells. The performance of the window-based method was found to be comparable to the modified method in the first test case. In addition to this quantitative performance enhancement, both of our methods employed post-processing steps which enhances the quality of the final clump splitted image.

The clump splitting methods developed here as well as most of the other methods in the literature operate on the binary segmented image. None of the methods exploit the intensity values of the gray-scale image to help in finding the split lines. One possible direction of future work is to find the split line between two concavity points following a minimum intensity path instead of making a straight line between them. This may also help in getting correct split lines for the cases in which there is a hole, formed due to clustering of objects, lying inside the boundaries of the clumped objects. Apart from these issues, development of completely new methods for fast detection of all the valid concavity points is also a possible extension of this work in the future.

References

[1] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*, CRC Press, 1997.

[2] X. Bai, C. Sun, and F. Zhou, "Splitting touching cells based on concave points and ellipse fitting," *Pattern Recognition*, vol. 42, no. 11, pp. 2434-2446, November 2009.

[3] D. Balthasar, T. Erdmann, J. Pellenz, V. Rehrmann, J. Zeppen, and L. Priese, "Real-time detection of arbitrary objects in alternating industrial environments," in *Proceedings of 12th Scandinavian Conference on Image Analysis*, pp. 321-328, June 2001.

[4] S. Beucher and C. Lantejuol, "Use of watersheds in contour detection," in *Proceedings of the International Workshop on Image Processing: Real-time edge and motion detection/estimation*, September 1979.

[5] J. Canny, "A computational approach to edge detection," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, November 1986.

[6] G. Cong and B. Parvin, "Model-based segmentation of nuclei," *Pattern Recognition*, vol. 33, no. 8, pp. 1383-1393, August 2000.

[7] S. Eddins, "Cell Segmentation," http://blogs.mathworks.com/steve/2006/06/02/cell-segmentation, Visited 23.10.2009.

[8] M. Farhan, O. Yli-Harja, and A. Niemistö, "An improved clump splitting method for convex objects," in *Proceedings of the 7th International Workshop on Computational Systems Biology*, pp. 35-38, June 2010.

[9] T. Fawcett, "An Introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, June 2006.

[10] G. Fernandez, M. Kunt, and J-P. Zryd, "A new plant cell image segmentation algorithm," in *Proceedings of 8th International Conference of Image Analysis and Processing*, pp. 229-234, September 1995.

[11] A. Fitzgibbon, M. Pelu, and R.B.Fisher, “Direct Least Sqaure Fitting of Ellipses,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476-480, May 1999.

[12] K. Goatman, “Automated detection of microaneurysms,” http://www.biomed.abdn.ac.uk/Abstracts/A07890, Visited 10.08.2010.

[13] R.C. Gonzalez and R.E. Woods, *Digital image processing*, Prentice Hall, second edition, 2002.

[14] G. Guo, X. Ping, D. Hu, and J. Yang, “An efficient segmentation algorithm based on mathematical morphology and improved watershed,” *Intelligent Computing in Signal Processing and Pattern Recognition, Lecture notes in control and information sciences*, vol. 345, pp.689-695, 2006.

[15] X. C. He and N.H.C Yung, “Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support,” in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2, pp. 791-794, August 2004.

[16] X. C. He and N.H.C Yung, “Corner detector based on global and local curvature properties,” *Optical Engineering*, vol. 47, no. 5, pp. 057008-1-12, May 2008.

[17] S. Kothari, Q. Chaudry, and M.D. Wang, “Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques,” in *Proceedings of the 6th IEEE International Conference on Symposium on Biomedical Imaging: From Nano to Macro*, pp. 795-798, 2009.

[18] S. Kumar, S.H. Ong, S. Ranganath, T.C. Ong, and F.T. Chew, “A rule-based approach for robust clump splitting,” *Pattern Recognition*, vol. 39, no. 6, pp. 1088-1098, June 2006.

[19] J. Liang, “Intelligent splitting in the chromosome domain,” *Pattern Recognition*, vol. 22, no. 5, pp. 519-532, 1989.

[20] N. Lu, X. Ke, “A segmentation method based on gray-scale morphological filter and watershed algorithm for touching objects image,” in *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Delivery*, vol. 3, pp. 474-478, 2007.

[21] A. Niemistö, T. Aho, H. Thesleff, M. Tiainen, K. Marjanen, M-L. Linne, and O. Yli-harja, ”Estimation of population effects in synchronized budding yeast experiments,” in *Proceedings of the International Society for Optical Engineering*,

SPIE 2003. Image Processing: Algorithms and Systems II, vol. 5014, pp. 448-459, 2003.

[22] A. Niemistö, J. Selinummi, R. Saleem, I. Shmulevich, J. Aitchison, and O. Yli-Harja, ”Extraction of the number of peroxisomes in yeast cells by automated image analysis,” in *Proceedings of the Engineering in Medicine and Biology Society, 28th Annual International Conference of the IEEE*, pp. 2353-2356, August- September 2006.

[23] N. Otsu, “A threshold selection method from graylevel histograms,” *IEEE Transactions on systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62-66 January 1979.

[24] S. Raman, B. Parvin, C. Maxwell, and M.H.Barcellos-Hoff, ”Geometric approach to segmentation and protein localization in cell cultured assays,” *Journal of Microscopy*, vol. 225, no. 1, pp. 22-30, 2007.

[25] A. Rosenfeld, “Measuring the size of concavities,” *Pattern Recognition Letters*, vol. 3, no. 1, pp. 71-75, January 1985.

[26] C.D. Ruberto, A. Dempster, S. Khan, and B. Jarra, “Segmentation of blood images using morphological operators,” in *Proceedings of the 15th International Conference on Pattern Recognition*, pp. 397-400, 2000.

[27] J.C Russ, *The Image Processing Handbook*, CRC Press and IEEE Press, third edition, 1999.

[28] T.L. Saito, M. Ohtani, H. Sawai, F. Sano, A. Saka, D. Watanabe, M. Yukawa, Y. Ohya, and S. Morishita, ”SCMD: Saccharomyces cerevisiae morphological database,” *Nucleic Acids Research*, vol. 32, pp. D319-D322, 2004.

[29] O. Schmitt and M. Hasse, “Radial symmetries based decomposition of cell clusters in binary and gray level images,” *Pattern Recognition*, vol. 41, no. 6, pp. 1905-1923, June 2008.

[30] L. Shen, X. Song, M.Iguchi, F. Yamamoto, “A Method for recognizing particles in overlapped particle images,” *Pattern Recognition Letters*, vol. 21, no., pp. 21-30, January 2000.

[31] I-M. Sintorn, *Segmentation Methods and Shape Description in Digital Images: Applications in 2-D andn 3-D Microscopy*, Swedish University of Agricultural Sciences, 2005.

[32] P. Soille, *Morphological Image Analysis: Principles and Applications*, second edition, Heidelberg, Germany: Springer Verlag, 2003.

[33] Y. Sun, S. Duthaler, and B.J. Nelson, "Autofocusing in computer microscopy: selecting the optimal focus algorithm," *Microscopy Research and Technology*, vol. 65, no. 3, pp. 139-149, October 2004.

[34] N. Sweeney and B.V. Sweeney, "Efficient segmentation of cellular images using Gradient-based methods and simple morphological filters," in *Proceedings of the Engineering in Medicine and Biology Society, 19th Annual International Conference of the IEEE*, pp. 880-882, October- November 1997.

[35] R.J. Taylor, D. Falconnet, A. Niemistö, S.A. Ramsey, S. Prinz, I. Shmulevich, T. Galitski, and C.L. Hansen, "Dynamic analysis of MAPK signaling using a high-throughput Microfluidic single-cell imaging platform," in *Proceedings of National Academy of Science USA*, vol. 106, no. 10, pp. 3758-3763, 2009.

[36] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583-598, June 1991.

[37] W. Wang and H. Sung, "Cell Cluster Image Segmentation on Form Analysis," in *Proceedings of 3rd International Conference of Natural Computation*, pp. 833-836, August 2007.

[38] W.X. Wang, "Binary image segmentation of aggregates based on polygonal approximation and classification of concavities," *Pattern Recognition*, vol. 31, no. 10, pp. 1503-1524, 1998.

[39] Q. Wen, H. Chang, and B. Parvin, "A Delaunay triangulation approach for segmenting clumps of nuclei," in *Proceedings of 6th IEEE International Symposium on Biomedical Imaging*, pp. 9-12, June-July 2009.

[40] C. Wählby, *Algorithms for Applied Digital Image Cytometry*, Uppsala University of Sweden, 2003.

[41] C. Wählby, J. Lindblad, M. Vondrus, E. Bengtsson, and L. Björkesten, "Algorithms for cytoplasm segmentation of fluorescence labeled cells," *Analytical Cellular Pathology*, vol. 24, no. 2-3, pp. 101-111, 2002.

[42] G. Zhang, D.S. Jayas, N.D.G. White, “Separation of touching grain kernels in an image by ellipse fitting algorithm,” *Biosystems Engineering*, vol. 92, no. 2, pp. 135-142, October 2005.

[43] Q. Zhong, P. Zhou, Q. Yao, and K. Mao, “A novel segmentation algorithm for clustered slender-particles,” *Computer and Electronics in Agriculture*, vol. 69, no. 2, pp. 118-127, December 2009.